Assembly Line

Alejandro Millan, Joshua Pena, and Harsh Bhakta

***Abstract*— In todays world, the automobile industry almost entirely relies upon assembly lines as a primary method of large-scale production. The process usually consists of state-of-the-art robotics along with intricate software that is customized as per the the manufacturer's needs. However, despite the exceptional advances that have been made in automobile production, machine failures continue to remain an issue. Machine breakdowns are detrimental to the throughput of automobiles, resulting in a potential loss of profit for manufacturers. This paper discusses a proposition in order to address this problem such that the throughput level can be maintained above the zero mark. The goal is to maintain production regardless of any functional mishaps that may occur. The algorithm suggested does not fully restore the level of throughput, but rather has production operate at less than the optimal level.**

I. INTRODUCTION

A. Background

Throughput is the key performance measure in an assembly line. Automobile companies need to reach their target throughput as planned. If the target isnt reached, the company will lose customers as well as money. The largest threat to production on assembly lines is the breakdown of robots. So, the best way to optimize production is to work on preventing machine breakdown[1].Though breakdown is unavoidable, there must be solutions that attempt to deal with the breakdown of machines. There has been many literature papers that are published in respect to machine breakdown. Ilar researched the effects on productivity when a new machine replaces an old one[2]. There has been work done surrounding the production of the assembly line. For example, there has been work done to focus on the optimal path of a single machine when given multiple tasks to complete [6]. Additionally, there has also been work done to focus on the optimal placement of machines in a factory [4]. These examples serve as ways to optimize an assembly line such that they are more efficient. While we are also working toward improving the assembly line, our goal is to prevent the production from halting rather than optimizing the assembly line.

B. Overview

Our project will try to demonstrate the production slow down effect by having one robot shut down in an assembly line production. Then we will simulate and compare the effects of having a different robot take over the powered down robot's task. This project will try to extend the efficiency of assembly line robots by having robots take over the tasks of the shut down robot. Ideally, if a robot were to fail, there will be an autonomous communication to the cloud server. This wireless communication will allow the bots to start taking over the task of the broken robot without any human intervention. Though the production line is expected to decrease, we expect the throughput to be greater than zero.

To achieve this project goal we must be certain that the robots have an overlapping workspace. Since we cannot have one robot completely overlap another robot's workspace, we must find ways for the task to be completed by other robots. This could be done by either making the task achievable in the workspace of a different robot or by having the workspaces overlap such that another robot can reach the needed area.

Fig. 1. Overlapping Workspace

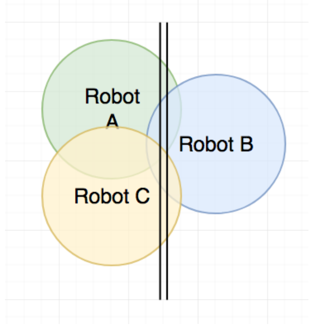


Figure 1 demonstrates the overlapping workspace the robots would have to have in order to completely take over a task.

C. Goal

The goal of the project is to demonstrate that the shutdown of assembly lines due to the breakdown of a machine can be prevented by taking advantage of other machine's workspaces to cover the task that the broken machine was responsible for.

D. Requirements

In order to ensure that the production line will run smoothly, there are a few requirements that must be met.

1) The malfunctioning robot must get out of the way of the assembly line and return to its idle position to avoid collision with another robot.
2) Each robot has the ability to communicate with the network in order to check if it has to take on an additional task.
3) Speed of production is automatically adjusted to allow enough time for the functioning robots to take over a task.
4) There exists multiple machines with similar tool tips such that a robot can easily take over the task of a malfunctioning robot
5) Each robot has similar tool tips to perform the same task as each robot in the production line

E. Limitations

Though the workspaces of the robots may overlap, there are still limitations to what a robot can take over. For example, the orientation of the robot matters in the task. Though the robots would be able to reach in the same location as one another, if the robot cannot perform the task in the correct orientation then the robot would fail to take over the task. Another limitation is in the case the robot at the end of the production line fails. Since the robots at the beginning and end of the assembly line have workspaces outside the range of all the other robots, the production line may stop completely or result in one bad product.

To help increase the chances of a robot taking over another robot's task, it is beneficial for the robot to have more number of joints. This gives the robot more freedom to be in different angles and orientations in order to be able to complete tasks that might not otherwise have been completed.

II. Simulation Setup

In our assembly line simulation, the robots are assigned to stack boxes that are 3 units high. The simulation we created has two parts. The first part is the VREP part which simulates the robots and the packages while the second part consists of python scripts that serve as a communication from the network to the robots.

A. VREP

Fig. 2. VREP layout

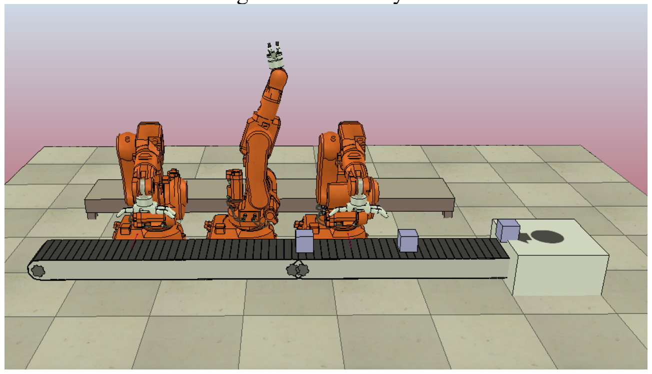


Figure 2 shows the layout view of our VREP simulation. The object in the far right produces packages until there is a box in front of each robot. Then it waits until all the boxes are cleared before pumping more boxes out. The boxes move via a conveyor belt. In this example, the robot in the middle is considered to be the ̈dead̈robot. As you can see the dead robot stands away from the conveyor belt in order to avoid collisions with the other robots while they grab the dead robot's box.

We are using the IRB140 robot that VREP provided along with it's template code. With Inverse kinematics, we were able to grab the box in front of the robot by using the following Cartesian coordinates: 0, 0.5, 0.5. This is because the conveyor belt is 0.5m in front of my robot and the box is elevated 0.5m from the floor. Since this robot has a maximum functional workspace of up to 0.6m radius, we decided to place each robot 0.6m away from the previous robot. This gives the robots enough space to prevent them from being in each other's way while still making it possible to have overlapping workspaces.

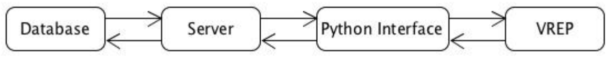
Originally, I had the robots move from the conveyor belt to the desk right away. However, the software did not choose the best path for carrying a package. Every time a robot picked up a package it would drop the package before getting to the destination. In order to correct this, I added an intermediate position. This intermediate path gave the robot a more structured way into getting to the destination.

I first attempted to have the robots stack 4 boxes, but when the robot tried to assist the malfunctioning robot, a singularity point was reach and the functional robot didn't have the proper orientation to drop the 4th box on top.

B. *Python*

We used Python to communicate between the network and VREP. Using python scripts, we were able to retrieve the necessary information from the server, calculate changes that may need to be made, and manipulate the machines in VREP. Figure 3 shows how the communication was set up. Information such as the robot's initial task that

Fig. 3. Network layout



was assigned, the placement of the robot in the assembly line and information about the the machines and the state of those machines were stored in a database. Once the information is processed using the python scripts,VREP is launched to show the visual representation. The network system was initially intended to be solely run through the server. The system was then changed to fit the Python aspect in. The server had a similar role, but instead of maintaining the database and processing the information, it just maintained the database. This way Python can make requests to the server, process the information, and communicate with VREP to show the results.

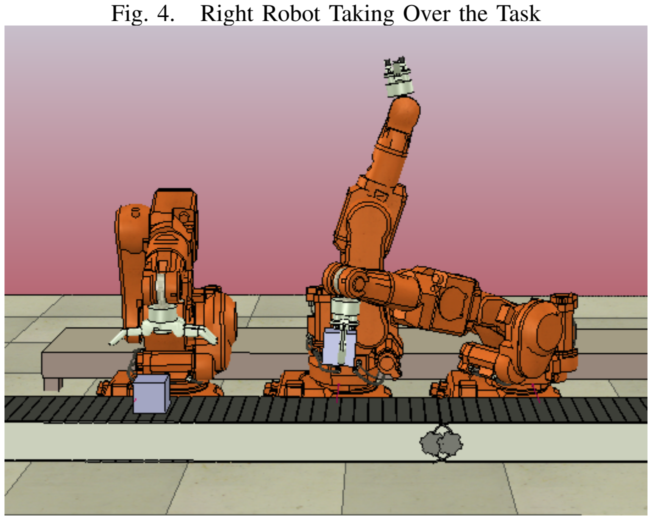
III. Conclusion

A. *Results*

When all robots are functional, the maximum throughput is achieved. The robots can stack 9 boxes in 32 seconds. If these robots were to operate 8 hours per day, then they could stack 8,100 boxes as shown in Eq.1, assuming that no failures occurred on that particular day. This means that the maximum throughput for a regular day (when the robots are in operation for 8 hours a day) is 8,100 boxes/day.

$$\frac{9boxes}{32s} * \frac{60s}{1min} * \frac{60min}{1hr} * 8hr = \mathbf{8{,}100} \tag{1}$$

Fig. 4. Right Robot Taking Over the Task



When we have one robot that starts to malfunction, one of the working robots need to take over. In Fig. 4, it is shown that the middle robot breaks down. We tested to see how the throughput would be like if the robot on the right took over the broken robot's task. We expected that our throughput would be half of the maximum throughput that was calculated from Eq.1. However, the experimental throughput turned out to be 9 boxes in 70 seconds.

$$\frac{9boxes}{70s} * \frac{60s}{1min} * \frac{60min}{1hr} * 8hr = \mathbf{3{,}703} \tag{2}$$

Therefore, when the middle robot is broken the entire day and the rightmost robot takes over its task, we can stack only 3,703 boxes. The throughput went down by 54%. This means in order to stack 8,100 boxes, it will take approximately 17

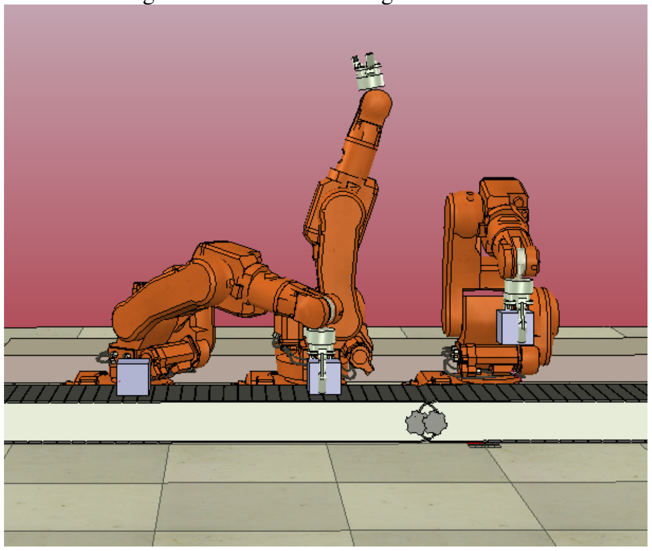
TABLE I

HOURS OPERATED VS BOXES STACKED

Hours Operated	Boxes Stacked	Loss
8	3,703	54%
12	5,554	31%
14	6,480	20%
16	7,406	9%
17.5	8,100	0%

hours and 30 minutes instead of 8 hours. Below in Table 1, is the total amount of boxes that can be stacked in the case the middle robot is broken and the rightmost robot takes over.

Fig. 5. Left Robot Taking Over the Job



The second scenario we tested is if the left robot takes over the broken robot's task as shown in Fig.5. We expected that our throughput would be half of the maximum throughput that we got from Eq.1. However, the experimental throughput turned out to be 9 boxes in 65 seconds. This is 5 seconds faster than the rightmost robot. This is because the rightmost robot has to take a longer path than the leftmost robot in order to take over the task of the broken robot.

$$\frac{9boxes}{65s} * \frac{60s}{1min} * \frac{60min}{1hr} * 8hr = \mathbf{3{,}987} \tag{3}$$

Therefore, when the middle robot is broken for the entire day, and the leftmost robot takes over, we can stack only 3,987 boxes. The throughput went down by 50% as we predicted for this robot. In order to stack 8,100 boxes, it will take 16 hours and 15 minutes. Below in Table 2, is the total amount of boxes that can be stacked in the case the middle robot is broken and the leftmost robot takes over for various hours operated.

TABLE II

HOURS OPERATED VS BOXES STACKED

Hours Operated	Boxes Stacked	Loss
8	3,987	50%
10	4,984	38%
12	5,981	26%
16	7,975	1%

TABLE III

HOURS OPERATED VS BOXES STACKED

Hours Operated	Boxes Stacked	Loss
8	4,712	41%
10	5,891	27%
12	7,069	13%
13.75	8,100	0%

In the third scenario was had the leftmost and rightmost robots alternate taking over the broken robot's task. We expected this scenario to be the best in terms of throughput. As a result of this setup, the experimental throughput turned out to be 9 boxes in 55 seconds. This is 10 seconds faster than the rightmost robot taking over the entire task and 5 seconds faster than the leftmost robot taking over the entire task.

$$\frac{9boxes}{55s} * \frac{60s}{1min} * \frac{60min}{1hr} * 8hr = \mathbf{4{,}712} \tag{4}$$

Therefore, when the middle robot is broken the entire day, and both robots alternate between taking over the broken robot's task, we can stack a total of 4,712 boxes. The throughput went down by 41%. This is much better than 54% and 50%, the throughputs resulting from the other scenarios. We were able to increase the throughput by 9% compared to the second scenario and by 13% compared to the first scenario. This means in order to stack 8,100 boxes, it will take approximately 13 hours and 45 minutes. Below in Table 3, is the total amount of boxes that can be stacked in the case the middle robot is broken and both the leftmost and rightmost robots take over.

After analyzing these three scenarios, we can clearly see that having both the leftmost and the rightmost robots contribute to taking over the task of the middle robot yields the highest throughput. Therefore, The idea of having robots cover for their broken comrades is possible. This would help assembly lines from completely stopping.

B. Future Works

To improve our proposed solution, we can use the works other people who have worked on this problem. Using the work of Shin and Zheng[6], we can find the shortest path for the robot that could complete the task the quickest in the assembly line. Using the work of Lee, Khoo, and Yin[4] we can optimize the placement of the robots such that they can have overlapping workspace. Other ways to improve the system is to possibly break tasks down into smaller parts so once a robot breaks down, the task gets split into multiple parts. This way, robots that take over can work on smaller tasks rather than the entire task. This way, the time a robot has to add on to their originally assigned workload is decreased than if the robot were to take over the entire task.

In order to have proof of concept, we are using three robots to pick and place boxes. Since this works, we can keep adding additional pick and place machines. We can also add machines with other functions. The caveat is that there needs to be multiple of the same types of machines in order to be covered in case of breakdown. If machines of different tool tips are added, the order of operations must be taken into account. For example, if a cover is placed over a section of the object, then part of the workspace is no longer available. In these particular case, the robot may not be able to complete their required tasks on time.

Another experiment that could be done is to see how different tool tips can affect the throughput. We could compare the throughput of picking up boxes with the throughput of soldering a box. Then we can compare how quickly the other robots could take over the task and complete it.

Lastly, a more complex examination on how the throughput is affected by the position of the malfunctioning robot in the assembly line can be done. Our experiment only examines the case in which the middle robot fails but we could compare those results with the case in which the first robot or last robot in the assembly line malfunctions instead.

APPENDIX

The simulation videos are uploaded on the Google Drive and have been shared.

ACKNOWLEDGMENT

We would like to thank our professor for supporting us and guiding us to get better quality results and simulations.

REFERENCES

[1] M. Ozkok, The effects of machine breakdown on hull structure production process, Scientia Iranica, Volume 20, Issue 3, 2013, Pages 900-908, ISSN 1026-3098.

[2] T. Ilar, Simulation of production linesthe importance of breakdown statistics and the effect of machine position, Int. J. Simul. Model, 4 (2008), pp. 176-185

[3] L. Poultney, Hey Tesla, how hard can it be to actually make a car?, WIRED, 01-May-2018. [Online]. Available: http://www.wired.co.uk/article/tesla-model-3-production-stock-problems-engineering.

[4] S. G. Lee, L. P. Khoo, and X. F. Yin, Optimising an Assembly Line Through Simulation Augmented by Genetic Algorithms, The International Journal of Advanced Manufacturing Technology, vol. 16, no. 3, pp. 220228, 2000.

[5] Regular API function list (by category), V-REP User Manual.

[6] K. Shin and Q. Zheng, Scheduling job operations in an automatic assembly line, Proceedings., IEEE International Conference on Robotics and Automation.

[7] Shin, F. A decision tool for assembly line breakdown action, Proceedings of the 2004 Winter Simulation Conference, pp. 11221127 (2004).