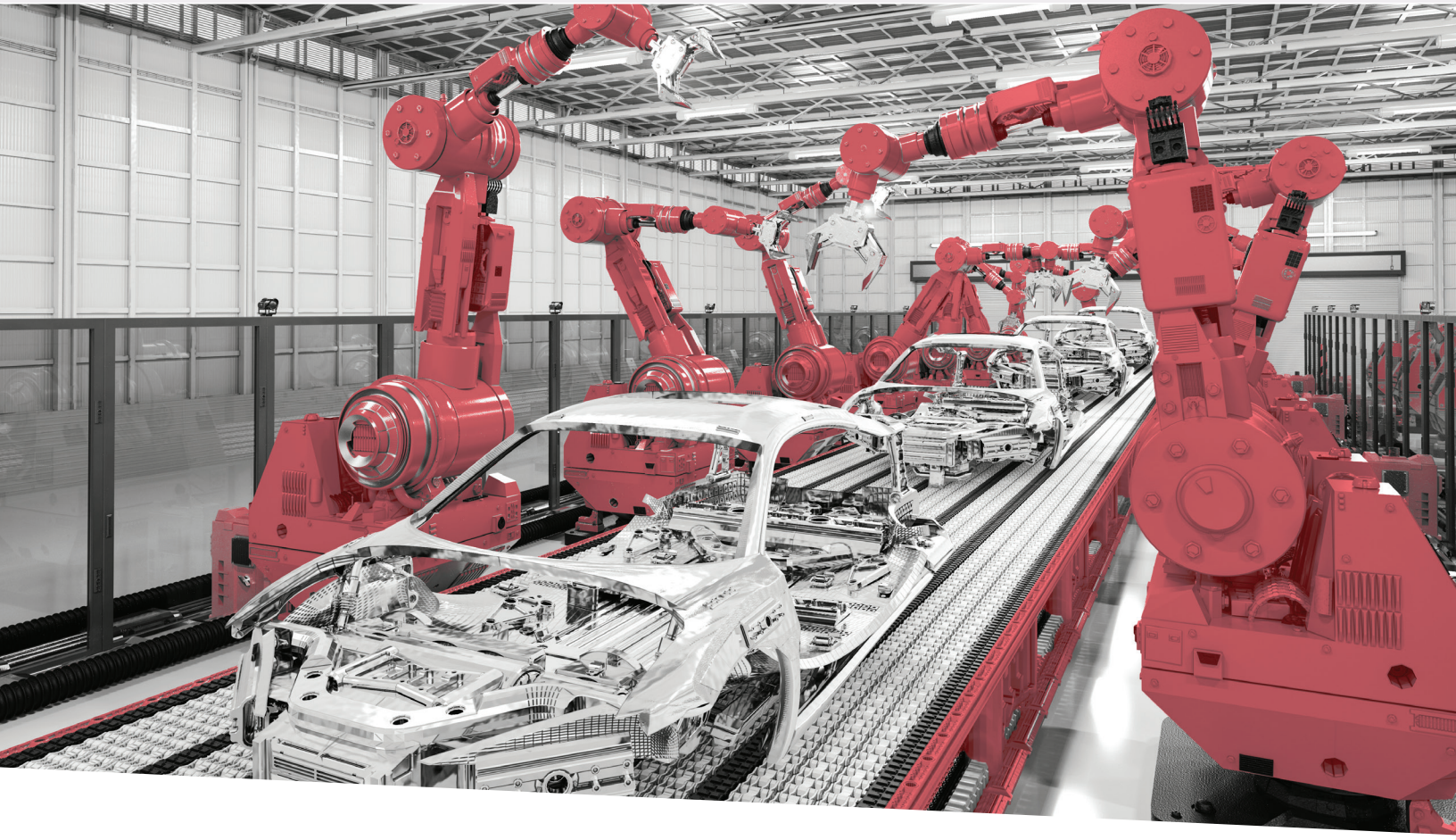


# *ADVANCING IMPLEMENTATION OF*
# MULTI-ROBOT WORKCELLS

There is a great opportunity to have multiple robots work together in many application domains, including manufacturing and logistics. The potential performance advantage of being able to use multiple robots is similar to the benefits of being able to use multiple people to complete a task.

realtime robotics

*27 Wormwood St Ste 110, Boston, MA 02210 | 617.302.6330 | rtr.ai | sales@rtr.ai*

# *DEPLOYMENT* PROCESS

**While many of us have seen multi-robot workcells, we do *not* see the vast amount of time required to deploy them and the performance opportunities that they miss.** Once the number of robots has been chosen, there are several challenging problems that must be solved, and they all involve choreographing the work. This work can be divided up into tasks (e.g., pick up object X or weld at location Y), and each task requires a robot to reach a target pose and spend some amount of time there performing an action. Engineers must solve the following interrelated challenges to achieve the best possible performance, often referred to as cycle time, without collisions:

## Placing

The location, position and angle of each robot and base should be placed where it can maximize its accessibility to its targets and tasks.

## Allocating

Every robot has a different set of capabilities (reach, function, tool-tip, etc.). Engineers determine which robot will accomplish which set of tasks, while understanding the limitations and complexities of each choice.

## Sequencing

This is the order and priority of each robot allocated to complete the given tasks.

## Robot Programming

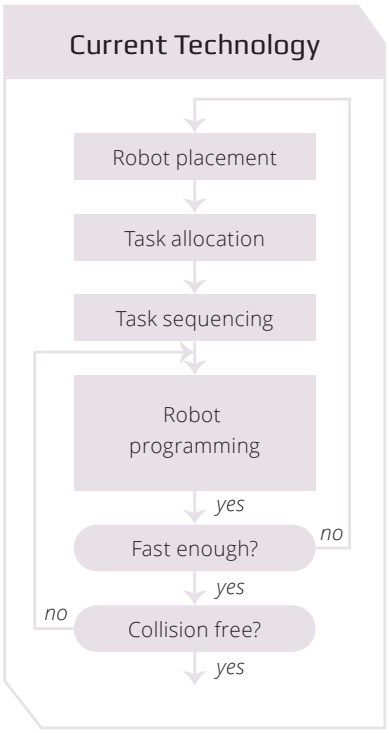
Once tasks have been allocated and sequenced, a program is written to instruct each robot where, when and which way to go.

## Avoiding Collisions

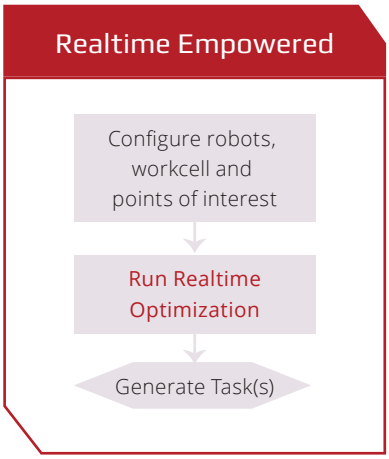
Programmers need to guarantee that robots avoid collisions while performing various tasks. Detecting and rerouting a robot to avoid collisions during a task has been computationally difficult and slow. Due to the complexity, programmers use conservative practices, such as pauses, interlocks and interference zones as a way to avoid collisions. Rather than using these tedious and performance hindering solutions, developers now have the option of using Realtime Robotics' technology.

# *REAL* PROGRESS

**Writing robot programs that provide collision-free motion plans is very hard, and producing optimized motion plans—including optimized placement, allocation and sequencing—is infeasible for engineers.** Current robot deployments require time-consuming processes to test and ensure that robots avoid all collisions. Due to the complexity of where multiple robots will be at any time, it is very difficult to produce collision-free plans. If there is a collision during testing (or worse during runtime), then an engineer modifies the robot program by either adding interlocks, pauses or changing the waypoints between targets. This time-consuming process continues until the robots pass all tests to be deployed.




The current process of modifying the software (displayed left) with iterative changes can degrade performance beyond what is acceptable, forcing the entire process to begin again from placement. It can require many weeks to produce even a viable collision-free solution; improving performance beyond the minimum viable has been prohibitively slow, and results have still been far from optimal.




**Realtime Robotics has developed a proprietary technology to quickly and autonomously produce highly optimized motion plans.** Realtime's technology drastically reduces engineering testing and development time required. Our toolkit allows users to intuitively add their virtual implementation, query tests and automatically output code. The code provided from the toolkit eliminates the need to manually write and iterate custom software. Realtime equips customers with a simple file for implementation rather than having weeks to months to produce less versatile results.

# *REAL* APPLICATION

***75%+ reduction in programming time/costs:***

A global auto manufacturer had a 4-robot workcell in which the robots needed to reach 22 targets and spend a half-second at each target to perform the desired task. Their automation engineer spent 13 weeks iterating to a cycle time of 21.4 seconds. Realtime's optimization technology achieved a cycle time of only 12.4 seconds, and it took only one week to achieve this result. With Realtime's solution, the robots were doing useful work 95% of the time, compared to only 75% of the time for the manual solution.

## *REAL* RESULTS

*Carried out by the Realtime Robotics optimization framework*

| Paramater | Customer Baseline | Realtime Result 1 | Realtime Result 2 | Realtime Result 3 |
|---|---|---|---|---|
| **Home poses** | Default | Default | Default | Optimal |
| **Robot placement** | Default | Default | Optimal | Optimal |
| **Task plan** | Default/Manual | Optimal | Optimal | Optimal |
| **Cycle time** | 21.43 seconds | 19.03 seconds | 13.87 seconds | 12.41 seconds |
| **Reduction** | -- | 11.20% | 35.27% | 42.09% |
| **Workcell efficiency** | 75.20% | 85.27% | 95.67% | 95..34% |
| **Total inactive time** | 21.26 seconds | 11.21 seconds | 2.4 seconds | 2.31 seconds |
| **Collision avoidance** | Pass* | Pass | Pass | Pass |
| **OLP time** | 13 weeks | 3 weeks | 1 week | 1 week |

*"Default" parameters result from the original project specification.*
*"Manual" parameters are generated by a manual tuning process.*
*"Optimal" parameters are the result of the optimization process .*
** Collision avoidance requires all robots to have accurate time synchronization*

realtime robotics

*27 Wormwood St Ste 110, Boston, MA 02210 | 617.302.6330 | rtr.ai | sales@rtr.ai*