Trajectory Planning and Comparison for a SCARA Robot Pick-and-Place Task

Ruohan He

Abstract— **This project investigates trajectory planning for a SCARA robot performing a pick-and-place task. Joint-space and task-space approaches are implemented and compared using identical via-point structures. Their kinematic behaviors, execution times, and trajectory characteristics are analyzed, highlighting trade-offs between efficiency, geometric fidelity, and joint-level optimization.**

I. Kinematic Modeling and Task Setup

A. DH Parameters

The robotic arm considered in this project is a four-degree-of-freedom SCARA manipulator consisting of two revolute joints in the horizontal plane, one prismatic joint for vertical motion, and one revolute joint for end-effector orientation control. All joint z-axes are defined to point upward. As a result, downward motion of the end effector corresponds to negative values of the prismatic joint displacement.

The DH parameters are defined according to the standard convention. The first two joints are revolute joints with link lengths L1 = 0.325 m and L2 = 0.225 m. The third joint is prismatic and controls the vertical position of the end effector, while the fourth joint controls the yaw orientation of the tool. The fixed offsets are d1 = 0.416 m and d4 = -0.080 m. Joint limits are imposed to reflect physical constraints of the robotic arm.

	Length	Twist	Offset	Angle
i	a_{i-1}	α_{i-1}	d_i	θ_i
1	0	0	d_1	θ_1
2	L_1	0	0	θ_2
3	L_2	0	d_3	0
4	0	0	d_4	θ_4

Figure 1. DH parameters

B. Forward kinematics

Using the defined DH parameters, the forward kinematics of the SCARA robot are derived by sequentially multiplying the homogeneous transformation matrices from the base frame to the end-effector frame. The resulting end-effector position in Cartesian space is expressed as:

$$x = L_1 \cos(q_1) + L_2 \cos(q_1 + q_2)$$
$$y = L_1 \sin(q_1) + L_2 \sin(q_1 + q_2)$$
$$z = d_1 + d_3 + d_4$$

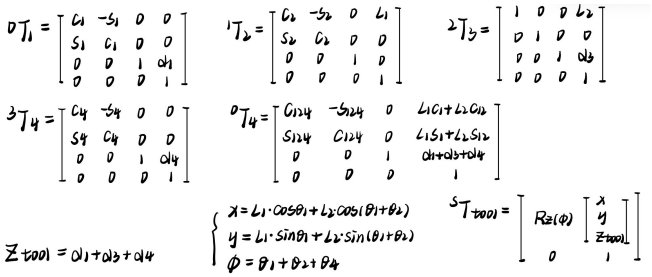

Figure 2. Forward kinematics

C. Inverse Kinematics

The inverse kinematics problem is solved analytically. Given a desired end-effector pose, the planar joint angles q1 and q2 are obtained using geometric relations, with both elbow-up and elbow-down solutions available. The prismatic joint displacement d3 is computed directly from the desired z-position, and the tool orientation joint q4 is determined from the desired yaw angle after accounting for q1 and q2.

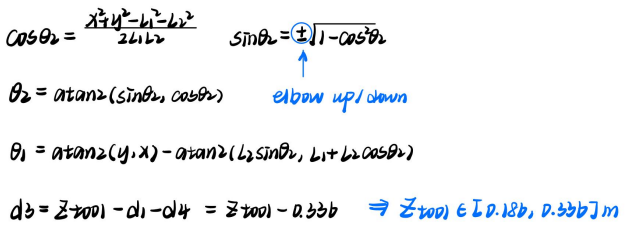

Figure 3. Inverse kinematics

D. Locations of the Feeder and PCB

A reference configuration is introduced to provide a consistent starting and ending pose for the pick-and-place task. The reference joint configuration is chosen as q_ref = [0, 90 , 0, 0], which places the robot in a compact, collision-free posture within the workspace. The corresponding reference pose is obtained through forward kinematics and is used as the initial and final pose of the overall trajectory.

The pick-and-place task consists of transferring four identical chips from a feeder to four target locations on a printed circuit board (PCB). Each chip has dimensions of 10 mm × 10 mm × 2 mm, and the PCB has dimensions of 100 mm × 100 mm. The chip placement locations are defined at the four corners of the PCB, offset inward by 5 mm so that the coordinates correspond to chip centers rather than PCB corners.

In the initial setup, the locations of the feeder and the PCB were chosen in an ad-hoc manner based on workspace reachability and geometric convenience. The feeder position

and PCB chip locations were specified directly in the base frame using fixed absolute coordinates, and no attempt was made at this stage to evaluate how these choices affected the overall execution time of the pick-and-place task. This initial configuration was therefore suitable for validating kinematic feasibility and collision avoidance, but it was not optimized with respect to trajectory duration.

In the final implementation, the station layout is parameterized by the PCB center location (x_P, y_P), which is set to (0.250, 0.225) m. The feeder position is defined relative to the PCB center by a fixed horizontal offset of 0.070 m in the positive x-direction, resulting in a feeder location of (x_F, y_F) = (x_P + 0.070, y_P). The four chip placement locations are generated automatically by offsetting the PCB center by ± 0.045 m in the x- and y-directions, which corresponds to a 100 mm × 100 mm PCB with 10 mm × 10 mm chips placed at the corners.

This parameterized formulation allows different feeder – PCB layouts to be explored efficiently by adjusting only the PCB center coordinates (x_P, y_P). By comparing the resulting total execution times produced by the trajectory planner, layouts leading to shorter overall motion durations can be identified.

```
T_F_pick (Feeder picking pose, w.r.t. {S})
[ 1.000000 -0.000000  0.000000  0.320000;
  0.000000  1.000000  0.000000  0.225000;
  0.000000  0.000000  1.000000  0.280000;
  0.000000  0.000000  0.000000  1.000000]

T_C1 (PCB target 1 pose, w.r.t. {S}) =
[ 0.000000  1.000000  0.000000  0.295000;
 -1.000000  0.000000  0.000000  0.180000;
  0.000000  0.000000  1.000000  0.280000;
  0.000000  0.000000  0.000000  1.000000]

T_C2 (PCB target 2 pose, w.r.t. {S}) =
[ 1.000000 -0.000000  0.000000  0.205000;
  0.000000  1.000000  0.000000  0.180000;
  0.000000  0.000000  1.000000  0.280000;
  0.000000  0.000000  0.000000  1.000000]

T_C3 (PCB target 3 pose, w.r.t. {S}) =
[ 0.000000 -1.000000  0.000000  0.205000;
  1.000000  0.000000  0.000000  0.270000;
  0.000000  0.000000  1.000000  0.280000;
  0.000000  0.000000  0.000000  1.000000]

T_C4 (PCB target 4 pose, w.r.t. {S}) =
[-1.000000 -0.000000  0.000000  0.295000;
  0.000000 -1.000000  0.000000  0.270000;
  0.000000  0.000000  1.000000  0.280000;
  0.000000  0.000000  0.000000  1.000000]
```

Figure 4. Locations expressed as homogeneous transformation matrices

E. Via Points and Height Selection

Via points are introduced to enforce a structured separation between horizontal transport, vertical insertion, and end-effector orientation changes. For the feeder and for each PCB target location, two poses are defined: a travel pose at a higher elevation and a pick/place pose at a lower elevation. Horizontal motion in the x – y plane and all yaw angle changes are performed exclusively at the travel poses, while vertical motion is restricted to transitions between the corresponding travel and pick/place poses.

In the initial design, the travel height and pick/place height were chosen as z_T = 0.24 m and z_P = 0.20 m, respectively. In the final implementation, these values are increased to z_T = 0.300 m for travel and z_P = 0.280 m for pick and place. Raising both the travel height and the pick/place height is primarily motivated by reducing the time spent on prismatic joint motion. By increasing the absolute operating heights, the required vertical displacement between consecutive pick-and-place actions is reduced from 0.04 m to 0.02 m, which shortens the duration of repeated prismatic joint movements and improves overall cycle efficiency, while still maintaining sufficient clearance above the feeder and PCB.

```
T_ref (w.r.t. {S}) =
[ 0.000000 -1.000000  0.000000  0.325000;
  1.000000  0.000000  0.000000  0.225000;
  0.000000  0.000000  1.000000  0.336000;
  0.000000  0.000000  0.000000  1.000000]

T_Ft (w.r.t. {S}) =
[ 1.000000 -0.000000  0.000000  0.320000;
  0.000000  1.000000  0.000000  0.225000;
  0.000000  0.000000  1.000000  0.300000;
  0.000000  0.000000  0.000000  1.000000]

T_Fp (w.r.t. {S}) =
[ 1.000000 -0.000000  0.000000  0.320000;
  0.000000  1.000000  0.000000  0.225000;
  0.000000  0.000000  1.000000  0.280000;
  0.000000  0.000000  0.000000  1.000000]

T_C1t (w.r.t. {S}) =
[ 0.000000  1.000000  0.000000  0.295000;
 -1.000000  0.000000  0.000000  0.180000;
  0.000000  0.000000  1.000000  0.300000;
  0.000000  0.000000  0.000000  1.000000]

T_C1p (w.r.t. {S}) =
[ 0.000000  1.000000  0.000000  0.295000;
 -1.000000  0.000000  0.000000  0.180000;
  0.000000  0.000000  1.000000  0.280000;
  0.000000  0.000000  0.000000  1.000000]

T_C2t (w.r.t. {S}) =
[ 1.000000 -0.000000  0.000000  0.205000;
  0.000000  1.000000  0.000000  0.180000;
  0.000000  0.000000  1.000000  0.300000;
  0.000000  0.000000  0.000000  1.000000]

T_C2p (w.r.t. {S}) =
[ 1.000000 -0.000000  0.000000  0.205000;
  0.000000  1.000000  0.000000  0.180000;
  0.000000  0.000000  1.000000  0.280000;
  0.000000  0.000000  0.000000  1.000000]

T_C3t (w.r.t. {S}) =
[ 0.000000 -1.000000  0.000000  0.205000;
  1.000000  0.000000  0.000000  0.270000;
  0.000000  0.000000  1.000000  0.300000;
  0.000000  0.000000  0.000000  1.000000]

T_C3p (w.r.t. {S}) =
[ 0.000000 -1.000000  0.000000  0.205000;
  1.000000  0.000000  0.000000  0.270000;
  0.000000  0.000000  1.000000  0.280000;
  0.000000  0.000000  0.000000  1.000000]

T_C4t (w.r.t. {S}) =
[-1.000000 -0.000000  0.000000  0.295000;
  0.000000 -1.000000  0.000000  0.270000;
  0.000000  0.000000  1.000000  0.300000;
  0.000000  0.000000  0.000000  1.000000]
T_C4p (w.r.t. {S}) =
[-1.000000 -0.000000  0.000000  0.295000;
  0.000000 -1.000000  0.000000  0.270000;
  0.000000  0.000000  1.000000  0.280000;
  0.000000  0.000000  0.000000  1.000000]
```

Figure 5. Via points expressed as homogeneous transformation matrices

F. A Model of The Robotic Arm

The robotic arm model rendered using the MATLAB Robotics Toolbox, as well as the extended MATLAB code for the entire project, are provided at the following link: https://github.com/hrh200398/RuohanHe_C263B/tree/main/Project_2

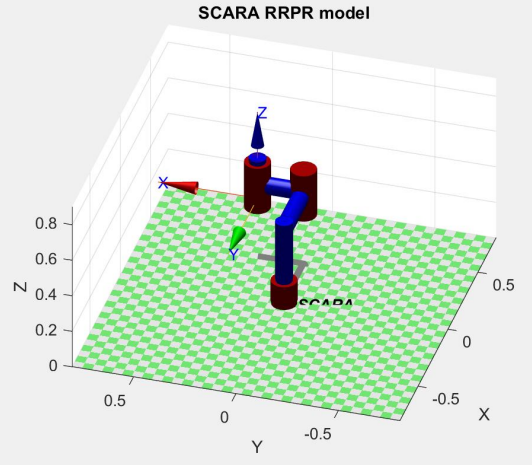

Figure 6. A model of the robotic arm

II. Trajectory Generation And Results

A. Implementation Overview

Two different trajectory generation approaches are implemented and evaluated in this project: a joint-space trajectory planning approach and a task-space trajectory planning approach. Both approaches share the same robot model, the same feeder and PCB geometric layout, and the same via-point structure defined in Part 1. The difference lies in how motion is interpolated between consecutive via poses and how kinematic constraints are enforced during trajectory generation.

B. Joint-Space Trajectory Generation and Results

In the joint-space approach, each via pose defined in task space is first converted into a joint-space waypoint using inverse kinematics with a fixed elbow configuration. Trajectory generation is then performed independently for each joint using a piecewise two-point linear-parabolic blend (LIPB) scheme subject to joint acceleration limits. The resulting trajectory achieves a near-minimum execution time of 22.21 s while strictly respecting joint-level kinematic constraints.

The effectiveness of the joint-space approach can be evaluated through the joint position, velocity, and acceleration plots. The joint position trajectories exhibit smooth transitions between via points, indicating continuity in position and velocity. The corresponding velocity profiles are piecewise continuous, while the acceleration profiles display characteristic step-like segments. These step changes in acceleration are a direct consequence of the LIPB formulation, in which constant-acceleration phases are concatenated with constant-velocity phases. Such acceleration profiles are expected and confirm that the trajectory is generated consistently with the intended joint-space blending method.

The three-dimensional end-effector trajectory generated by the joint-space approach does not form perfectly straight line segments between via points. This behavior is consistent with kinematic theory, since linear interpolation in joint space does not generally correspond to linear motion in Cartesian space. As multiple joints follow independent time laws, the resulting end-effector path appears curved in task space even when the start and end poses are geometrically aligned. Despite this deviation, the trajectory remains collision-free due to the conservative via-point design and the elevated travel height.

Overall, the observed joint-level and Cartesian behaviors are consistent with the theoretical properties of joint-space interpolation, indicating that the trajectory generation method is correctly implemented and behaves as expected.

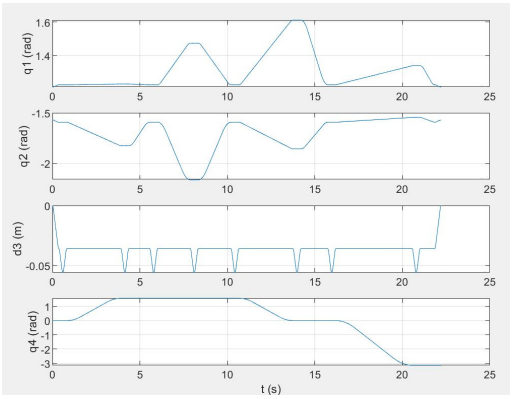

Figure 7. Joint position profiles generated by the joint-space approach

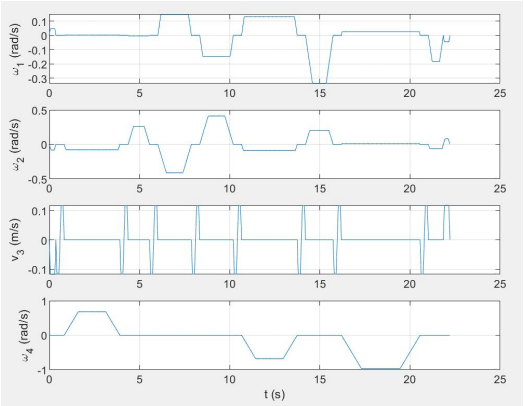

Figure 8. Joint velocity profiles generated by the joint-space approach

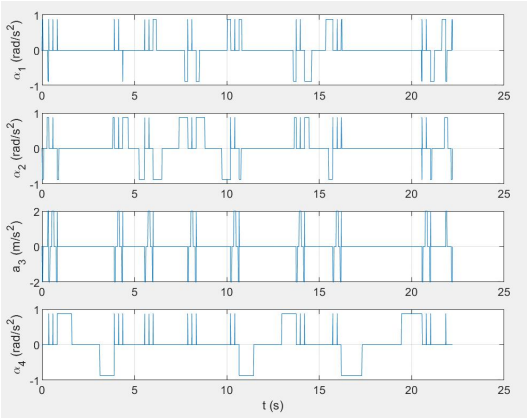

Figure 9. Joint acceleration profiles generated by the joint-space approach

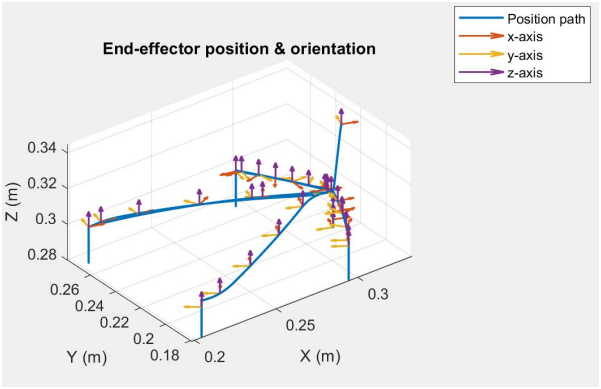

Figure 10. End-effector trajectory generated by the joint-space approach

C. *Task-Space Trajectory Generation and Results*

In the task-space approach, trajectories are generated directly in Cartesian space between consecutive via poses. The end-effector position and yaw angle are interpolated using a linear-segment-with-parabolic-blend scalar parameterization, producing smooth task-space position, velocity, and acceleration profiles. These task-space quantities are then mapped into joint space through the Jacobian and its time derivative.

The validity of the task-space approach is clearly reflected in the resulting end-effector paths. The three-dimensional position plots show nearly straight-line segments during horizontal transport at the travel height and strictly vertical segments during pick-and-place motions. This behavior arises because the interpolation is performed directly on the Cartesian position vector $p(t) = p0 + s(t)\,(p1 - p0)$, which enforces linear motion in task space by construction. Consequently, the geometric intent of the via-point layout is preserved throughout the entire trajectory.

The joint velocity and acceleration plots further demonstrate the effectiveness of the task-space method. Compared to the joint-space approach, the acceleration profiles are smoother and exhibit fewer abrupt changes. This is due to the adaptive time-scaling strategy, which increases segment durations whenever Jacobian-based acceleration limits are exceeded. Although this results in a longer total execution time of 30.71 s, it produces trajectories that more faithfully satisfy physical acceleration constraints.

These observations confirm that the task-space planner preserves the intended Cartesian motion primitives by construction, and that the resulting joint trajectories are a direct consequence of the imposed task-space constraints.

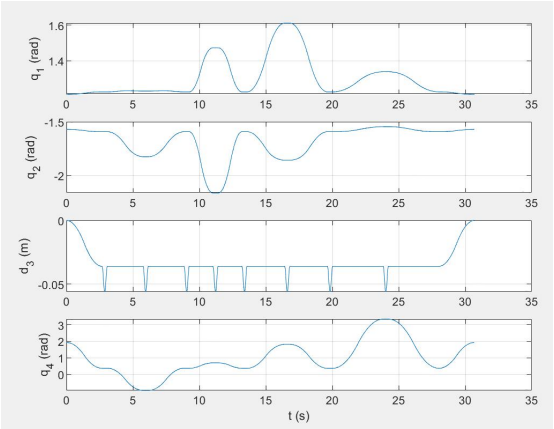

Figure 11. Joint position profiles generated by the task-space approach

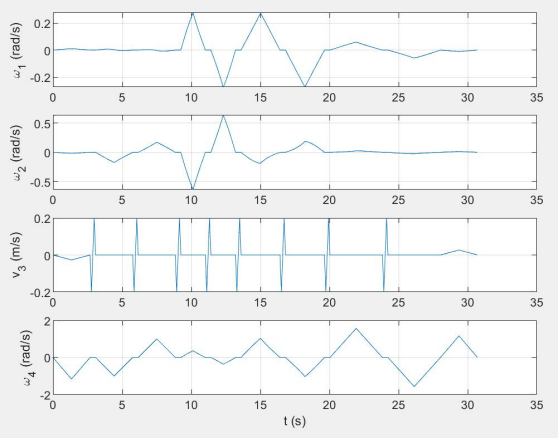

Figure 12. Joint velocity profiles generated by the task-space approach

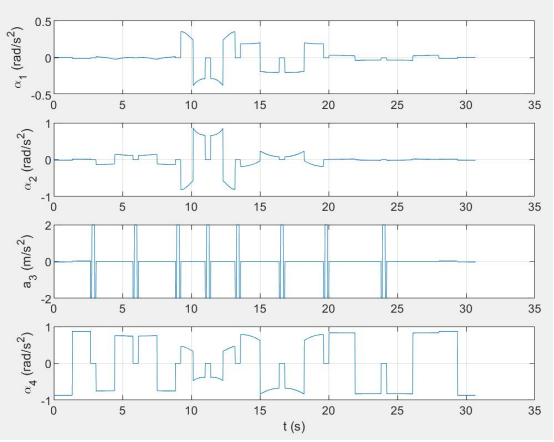

Figure 13. Joint acceleration profiles generated by the task-space approach

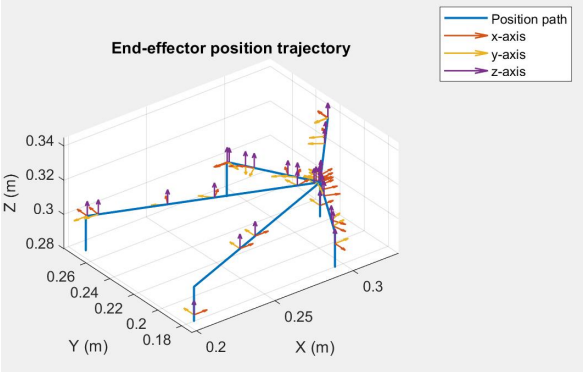

Figure 14. End-effector trajectory generated by the task-space approach

D. Verification of Design Conditions

In the joint-space approach, feasibility is ensured primarily by construction. The piecewise two-point LIPB formulation enforces joint acceleration limits explicitly, and the near-minimum segment durations are computed based on the most restrictive joint motion among all degrees of freedom. The resulting velocity and acceleration profiles exhibit the expected piecewise-constant and step-like characteristics, which are consistent with the underlying blending method and confirm that acceleration constraints are satisfied at all times.

In the task-space approach, feasibility is verified through Jacobian-based checks performed at every sampled time step. Joint accelerations are computed from the task-space accelerations using the analytical Jacobian and its time derivative, and the segment duration is adaptively increased whenever a joint acceleration limit is exceeded or when the Jacobian approaches singularity. The absence of acceleration limit violations in the final plots indicates that the adaptive time-scaling strategy is effective.

Collision avoidance is verified both implicitly and explicitly. Implicitly, the via-point structure guarantees that all horizontal motion and orientation changes occur at the travel height, while vertical motion is restricted to pick-and-place operations. Explicitly, the three-dimensional end-effector trajectories confirm that the robot maintains sufficient clearance above the feeder and PCB during horizontal transport and performs nearly vertical insertions during pick and place. Together, these observations demonstrate that the trajectory designs satisfy the imposed kinematic and safety constraints.

While both approaches satisfy feasibility and safety constraints, they differ in how efficiently these constraints are satisfied, which motivates the comparison discussed in the following section.

E. Discussion and Comparison

While both methods successfully satisfy the project requirements and maintain safety through collision avoidance, they exhibit distinct operational characteristics. The joint-space approach achieves a notably shorter execution time of 22.21 s, whereas the task-space approach requires a longer total duration of 30.71 s.

The differences in execution time are primarily driven by how each method enforces kinematic constraints and interpolates motion between via points. In the task-space approach, the end-effector position and yaw angle are interpolated directly in Cartesian space using a linear-segment-with-parabolic-blend parameterization, which enforces linear motion by construction. This ensures that the geometric intent of the via-point layout is preserved, resulting in straight-line transport and strictly vertical pick-and-place motions. Conversely, the joint-space approach utilizes a piecewise two-point linear – parabolic blend (LIPB) scheme applied independently to each joint. Because linear interpolation in joint space does not correspond to linear motion in Cartesian space, the resulting end-effector trajectory appears curved in task space. However, this deviation is acceptable as the trajectory remains collision-free due to the conservative via-point design and the elevated travel height.

The acceleration profiles further highlight the technical distinctions between the two methods. The joint-space method produces step-like acceleration segments, which are a direct consequence of the constant-acceleration phases inherent in the LIPB formulation. In contrast, the task-space approach generates smoother joint trajectories through the use of an adaptive time-scaling strategy. This strategy verifies feasibility via Jacobian-based checks at every time step and increases segment durations whenever acceleration limits are exceeded or the robot approaches a kinematic singularity. While this ensures high fidelity to physical constraints, it contributes to the significantly longer cycle time compared to the joint-space implementation.

For the specific pick-and-place task investigated in this project, the joint-space approach is identified as the superior solution. This preference is driven by its high efficiency and its inherent flexibility at the joint level, which allows for the selective suppression of unnecessary motions. Specifically, the tool orientation joint motion can be suppressed over path segments where end-effector yaw is not critical, thereby reducing the overall execution time without violating task requirements. These characteristics make the joint-space approach ideal for repetitive industrial operations where maximizing cycle time efficiency is a primary objective.