Optimization of a two-link Robotic Manipulator

Zachary Renwick, Yalım Yıldırım

April 22, 2016

Abstract

Although robots are used in many processes in research and industry, they are generally not customized for specific applications and thus are not optimal. In this work we focused on parameterizing an optimal SCARA robot for a simple pick and place application. Namely, we attempted to minimize the time needed for a robot's end effector to reach a set of waypoints distributed over a two dimensional space subject to the constraint of constant end effector velocity. To achieve this goal we developed a dynamic model of a simple two link robotic arm that represents the SCARA robot, and a model to fit a polynomial spline trajectory through an arbitrary set of points in a two dimensional plane. These models were implemented as MATLAB functions. Development of the optimization models for the robotic arm and trajectory planning were performed separately. Preliminary optimization results were obtained for each subsystem to validate the models and provide a proof of concept. Detailed descriptions of each subsystem and their preliminary results are presented. The subsystems were brought together at the system level and a combined optimal solution was found. Finally, a sensitivity analysis indicating the dependence of the solution on all applicable parameters and constraints is provided.

1 Introduction

The robot that we considered in this study uses the SCARA configuration, seen in Figure 1. A SCARA robot is generally defined as a robotic arm with revolute joints that span a 2-D horizontal plane. A dynamic two-link planar manipulator model was used to represent the robot, with the end effector position being defined at the tip of the second link. The mass of the links was assumed to be lumped at the joint positions, reflecting an assumption that link masses are negligible compared to the masses of actuators and end effectors placed at the joints. These assumptions allow inverse dynamics calculations to be easily adapted from a double-pendulum model [1]. The robot follows a trajectory that connects the waypoints that are considered for the problem. The waypoints represent a simple pick-and-place task such as picking products off an assembly line and placing them in a package. The trajectory is formed by performing a 2-D cubic spline fit on the waypoints. The robot's end effector is assumed to travel along the trajectory at constant speed. This assumption simplifies the problem by eliminating the need to optimize a velocity profile. Additionally, due to the nature of

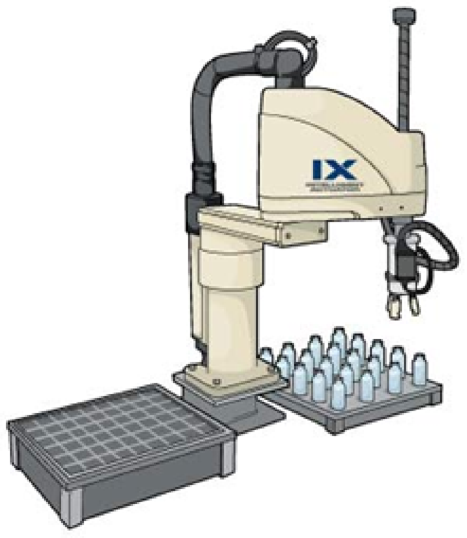
Figure 1. Pick and place robot utilizing a SCARA manipulator configuration

cubic splines this assumption means that the maximum accelerations will occur exactly at the waypoints. This fact simplifies the optimization problem by eliminating the need to check that acceleration and torque constraints are satisfied at every point on the trajectory. The robotic arm subsystem aims to find a robot configuration that minimizes the average torque applied during the execution of a given trajectory. The trajectory subsystem aims to find a trajectory that minimizes the time it takes to reach all waypoints under acceleration magnitude and constant velocity constraints. Shorter robotic link lengths minimize the inertia of the robotic arm and generally allow for faster acceleration, but reduce the robot's reachable work space and increase the angular distance through which each joint must rotate. Trajectories with short lengths often have abrupt changes in direction which require large accelerations and limit the speed at which they can be traversed. These two subsystems result in a non-obvious optimal robot and trajectory combination that accommodate these trade-offs while staying within constraints.

2 Subsection I: Robot

The goal of the robot optimization sub-problem is to find a robotic arm that can reach the entirety of a prescribed trajectory using minimal energy. This is accomplished by minimizing the applied torque averaged across both actuators and all trajectory waypoints. A robot with long links would have a large reachable space and would require lower joint velocities to achieve the same end effector speed. However, higher torques would be required to achieve the same angular acceleration since the mass elements would be far from the actuated joints. A small robot would have reachability problems and require higher joint velocities. The robot that uses minimal torque consumes the least energy and finds a good trade-off between these constraints.

Table 1. Robot Subsystem Optimization Formulation

Variables	Parameters	Objective	Constraints
L_1 (continuous)	m_1 (continuous)	$min\ mean(T_1, T_2)$	$T_1 < T_{max,1}$
L_2 (continuous)	m_2 (continuous)		$T_2 < T_{max,2}$
x_c (continuous)	$T_{max,1}$ (continuous)		$L_1 + L_2 > max\left((x_i^2 + y_i^2)^{\frac{1}{2}}\right)$
y_c (continuous)	$T_{max,2}$ (continuous)		$\lvert L_1 - L_2\rvert < min\left((x_i^2 + y_i^2)^{\frac{1}{2}}\right)$
			$L_1, L_2 > 0$

2.1 Problem Formulation

The design variables, parameters, objective and constraints are summarized in Table 1. The variables include the link lengths of the robot as well as the base position of the robotic arm. The parameters are the lumped masses of the two links and the maximum allowable active torques at the joints. The objective of this subsystem is to minimize the average of the torque values that the manipulator requires at instants it passes over the waypoints. Because the trajectory is formed using a cubic spline fit and a constant speed is assumed, the torques required at the waypoints are larger than other points on the trajectory. Thus by minimizing these torques, the torque profile along the trajectory is reduced. The constraints ensure that the actuators do not surpass their allowable maximum torques and the robot can reach all of the waypoints. The entire trajectory is not explicitly considered in the reachability constraint, but was confirmed post-analysis. From previous trials, testing only the waypoints seemed to be adequate for ensuring reachability for the entire trajectory.

2.2 Model

The physical model that was used for the robotic arm is shown in Figure 2. A double pendulum model with rigid links was incorporated as the model for the robotic arm. The end effector position is at the tip of the second link, and tracks the trajectory. L_1 and L_2 denote the length of the links, x_c and y_c the position of the robot's base, T_1 and T_2 the torque at the joints, m_1 and m_2 the lumped masses, and θ_1 and θ_2 the joint angles.

The numerical model is built by using Inverse Kinematics and Dynamics. The robot subsystem model assumes a trajectory. This trajectory is defined by the positions, velocities, and accelerations $[x, \dot{x}, \ddot{x}]$ required of the end effector at each waypoint. To find the required torques at each waypoint, the trajectory information is first converted to the generalized coordinate space, $[\theta, \dot{\theta}, \ddot{\theta}]$, using inverse kinematics. Although the robot's inverse kinematics have two solutions since it can reach any point using either an "elbow up" or "elbow down" configuration, it is constrained to use only one configuration to simplify the model. Then, Kane's Method is used to perform inverse dynamics and find the required actuator torques as shown in (1).

$$T = M(\theta)\ddot{\theta} - f(\theta, \dot{\theta}) \tag{1}$$

Here T is the generalized torque vector, M is the mass matrix that includes inertia information and f is the Coriolis and centrifugal forces that arise during robot motion. Gravitational

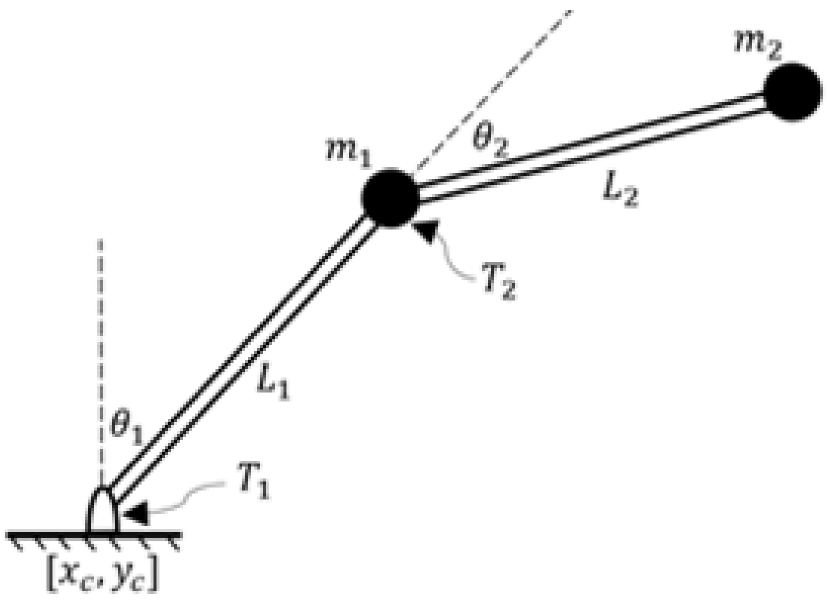

Figure 2. Schematic of the dynamics model used to represent the two-link robotic manipulator

Table 2. Robot Subsystem GA Optimization Parameters

Population size	1000
Number of Elites	10
Generations	25

forces are not included in the equation as the robot motion is assumed to take place in the horizontal plane.

2.3 Optimization Technique

A genetic algorithm using the parameters given in Table 2 was used to optimize the robotic arm subsystem. To improve the optimizer's performance, it was provided with an initial population in which all members were feasible solutions. The algorithm took approximately 15 minutes to converge to an optimal solution. This solution was confirmed through multiple subsequent runs of the optimization algorithm using randomly generated feasible initial populations. Gradient based optimization methods were initially attempted, but either did not converge to a solution or converged to a local minimum. The poor performance of gradient based methods can be explained by the non-convex design space and the existence of many local minima.

2.4 Results

The optimal solution for the robot sub-system is given in Table 3. The optimal robot and the trajectory for which it was optimized can be see in Figure 3. The robot was optimized to complete this trajectory in a time of 35 seconds. The masses m_1 and m_2 were both chosen to be equal to 1 kg, and both actuator torque limits were set to $T_{max} = 10$ N-m.

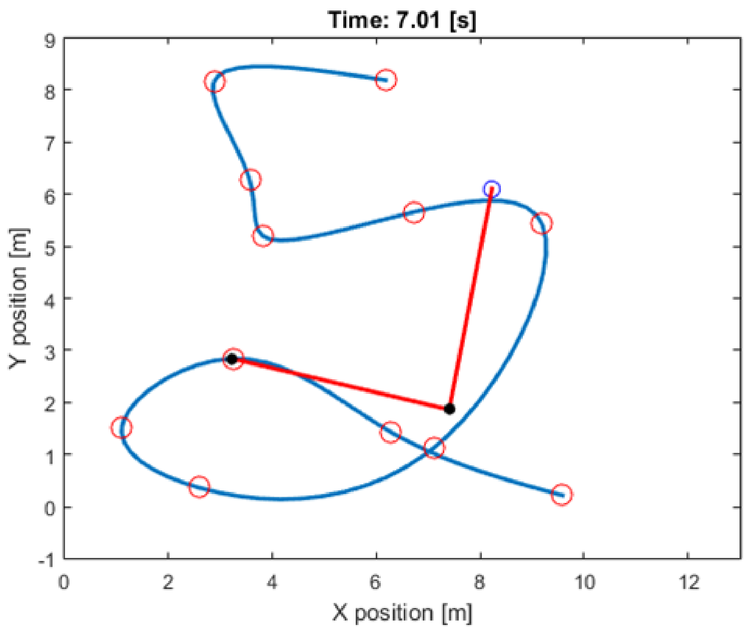

Figure 3. Optimal robot configuration for the robot subsystem

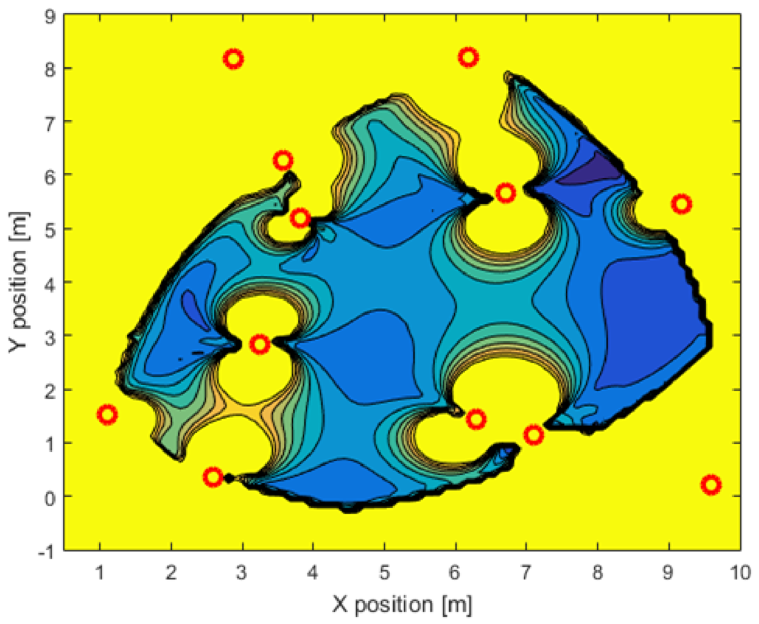

Figure 4. Objective function values of different base locations at optimal link lengths

Table 3. Robotic arm subsystem optimization results

Base position $\mathbf{x_c^*}, \mathbf{y_c^*}$ [m]	Link lengths $\mathbf{L_{1,2}^*}$ [m]	Mean torque T_{mean} [N-m]
$[8.23, 6.10]$	$[4.33, 4.30]$	0.9778

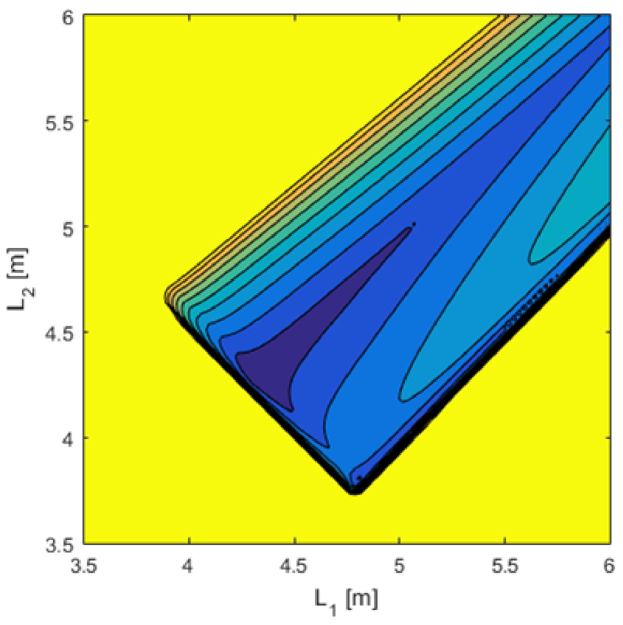

Figure 5. Objective function values of different link lengths at optimal base location

Figures 4 and 5 show cross sections of the objective function around the optimal solution through the x_c, y_c and L_1, L_2 planes respectively. The yellow region indicates designs where the reachability constraint is violated and no feasible solutions exist. This constraint makes the design space highly non-convex and not appropriate for gradient based optimization methods. Even with the genetic algorithm, the algorithm parameters had to be fine tuned to ensure convergence to a consistent solution. From Figure 5 it can be confirmed that the torque values are minimized when the links are chosen to be short. However, the shortest reachable solution will result in a kinematic singularity (axial alignment of both links) that limits the directions in which the tool tip can be accelerated. The optimizer converges to link lengths which are short enough to avoid excessive link inertia but long enough to avoid operating near the singularity point.

3 Subsection II: Trajectory

The goal of the trajectory optimization sub-problem is to find a trajectory which can visit a set of target points in minimal time. Any real dynamic system will be subject to acceleration constraints resulting from structure and actuator based limitations. Although such acceleration constraints are generally time, configuration, and direction dependent, they are modeled here as a bound on the tool tip acceleration magnitude. The model developed in subsection 3.2 using this constraint indicates that an optimal trajectory will balance the minimization of path length and the maximization of path smoothness as would be intuitively expected.

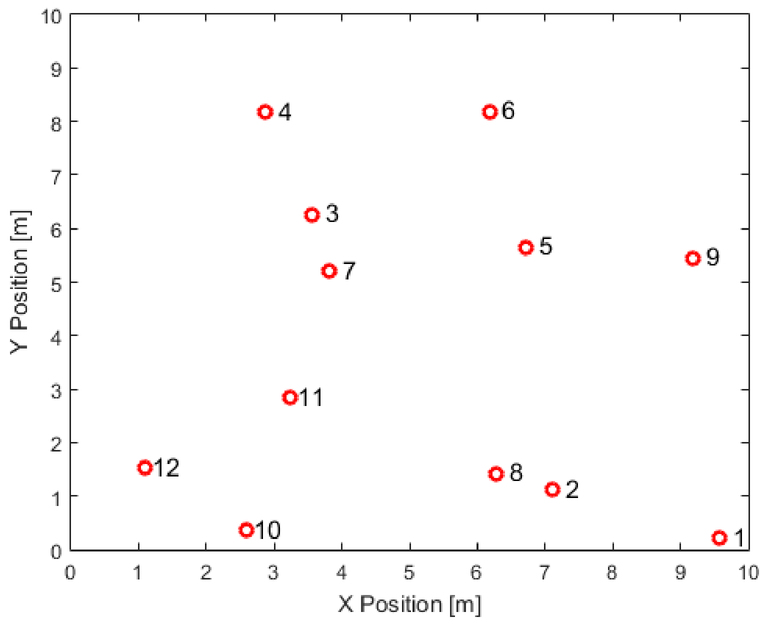

Figure 6. Randomly chosen trajectory target points with their numerical labels

Table 4. Trajectory Subsystem Optimization Formulation

Variables	Parameters	Objective	Constraints
p_i $i = 1..N_p$ (discrete)	$[x_i, y_i]$ $i = 1..N_p$ (continuous) a_{max} (continuous)	$min\ t_{path}$	$\|a_{tip}\| < a_{max}$ $d\|\mathbf{v}(t)\|/dt = 0$ $p_i \neq p_j$ for $i \neq j$

3.1 Problem Formulation

The design variables, parameters, objective and constraints for the trajectory sub-system optimization problem are summarized in Table 4. For a trajectory problem with N_p target points there are N_p discrete design variables each with N_p possible values. The constraint is imposed that each target point be visited exactly once. In addition to the acceleration magnitude constraint, the constraint is imposed that the tool tip must move at a constant speed throughout the trajectory. The constant speed constraint constraint both simplifies the problem under consideration and is a realistic requirement of many robotic applications. For this investigation, $N_p = 12$ points were randomly chosen from a $10m \times 10m$ area. The locations of these target points along with the numerical labels assigned to them can be seen in Figure 6. The value for a_{max} was arbitrarily chosen to be $10m/s^2$. In order to simplify the problem solution space and make the constraint that each point be visited exactly once implicit, all $N_p!$ permutations of the vector $x = [1, 2, ..., N_p]$ were associated with unique identification numbers. This change was implemented using the `nthperm` function from MATLAB's File Exchange. With this modification, the problem is reduced to a one dimensional optimization problem with a design variable who's domain is the set of integer values from the range $[1, N_p!]$.

3.2 Model

The ordered set of target points was converted into a smooth, continuous trajectory by performing a two dimensional cubic spline fit on the points. MATLAB's `cscvn` function was used to perform this spline fit. The resulting spline visits the target points in the order specified. However, finding the minimum time required to follow a trajectory subject to the constant velocity and acceleration constraints is itself an optimization problem. Fortunately, an analytical solution is possible for this problem. The equation for the trajectory is given by (2)

$$\begin{aligned} x &= f_x(s) = x(s) \\ y &= f_y(s) = y(s) \end{aligned} \tag{2}$$

where s is a parametric variable indicating the position along the length of the trajectory. Taking the time derivative of these equations using the chain rule results in (3).

$$\begin{aligned} \frac{dx(s)}{dt} &= \frac{dx(s)}{ds}\frac{ds}{dt} \\ \frac{dy(s)}{dt} &= \frac{dy(s)}{ds}\frac{ds}{dt} \end{aligned} \tag{3}$$

Finally, a second time derivative is taken to obtain the acceleration. The result shown in (4) is obtained by again applying the chain rule. The constant velocity constraint dictates that $\frac{d^2s}{dt^2} = 0$.

$$\begin{aligned} \frac{d^2x(s)}{dt^2} &= \frac{dx^2(s)}{ds^2}\frac{ds}{dt} \\ \frac{d^2y(s)}{dt^2} &= \frac{dy^2(s)}{ds^2}\frac{ds}{dt} \end{aligned} \tag{4}$$

Since only the magnitude of the acceleration is of interest in this case, the acceleration constraint can be expressed using (5).

$$|\mathbf{a_{tip}}| = \left(\frac{d^2x}{dt^2}^2 + \frac{d^2y}{dt^2}^2\right)^{\frac{1}{2}} = \left(\frac{ds}{dt}\right)^2 \left(\frac{d^2x}{ds^2}^2 + \frac{d^2y}{ds^2}^2\right)^{\frac{1}{2}} \tag{5}$$

Since $\frac{ds}{dt}$ is the constant velocity at which the trajectory is traversed at, it can be represented as $\frac{\Delta s}{\Delta t} = \frac{l_{path}}{t_{path}}$ where l_{path} is the total length of the trajectory and t_{path} is the time it takes to execute the path. Plugging this term into (5) then yields (6).

$$|\mathbf{a_{tip}}| = \left(\frac{l_{path}}{t_{path}}\right)^2 \left(\frac{d^2x}{ds^2}^2 + \frac{d^2y}{ds^2}^2\right)^{\frac{1}{2}} \tag{6}$$

A basic monotonicity analysis shows that the acceleration constraint must be active in order to bound the minimization of path time t_{path}. Consequently, the acceleration constraint will be satisfied equally so $|\mathbf{a_{tip}}| = a_{max}$ when t_{path} is minimized and the path curvature is at its maximum value. Plugging in this result and solving for the minimum path time yields (7).

Table 5. Trajectory Subsystem GA Optimization Parameters

Population size	10,000
Number of Elites	1000
Max Generations	100

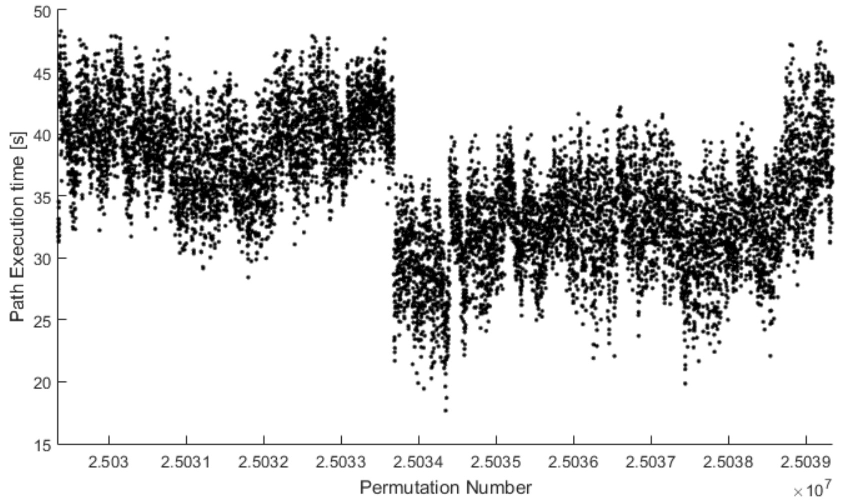

Figure 7. Sample of the design space for the trajectory optimization problem

$$t_{path} = \left(\frac{l_{path}^2}{a_{max}} max\left(\left(\frac{d^2x_i}{ds^2}^2 + \frac{d^2y_i}{ds^2}^2 \right)^{\frac{1}{2}} \right) \right)^{\frac{1}{2}} \tag{7}$$

This equation finds the minimal time that a trajectory can be executed in while implicitly satisfying the constant velocity and acceleration constraints. As a result, (7) serves as the objective function for an unconstrained formulation of the trajectory optimization problem.

3.3 Optimization Technique

Given the discrete, unconstrained nature of the reformulated trajectory optimization problem a genetic algorithm was chosen to search for a minimum. The objective function created a trajectory matching the prescribed target point sequence and determined the minimum path execution time from it using (7). MATLAB's `fnder` function was used to find derivatives of the spline with respect to s. Table 5 gives the genetic algorithm parameters used for trajectory optimization. The genetic algorithm was initialized with a population of feasible solutions generated using MATLAB's `randi` function. Multiple runs of this optimizer were performed and top solutions from past trials were included in the initial population of subsequent iterations. A sample of the trajectory optimizer's design space is shown in Figure 7. Given the size and highly non-linear nature of the design space, a large population size was needed for the genetic algorithm to produce high quality solutions. As a result, each run of the optimizer required approximately 6 hours.

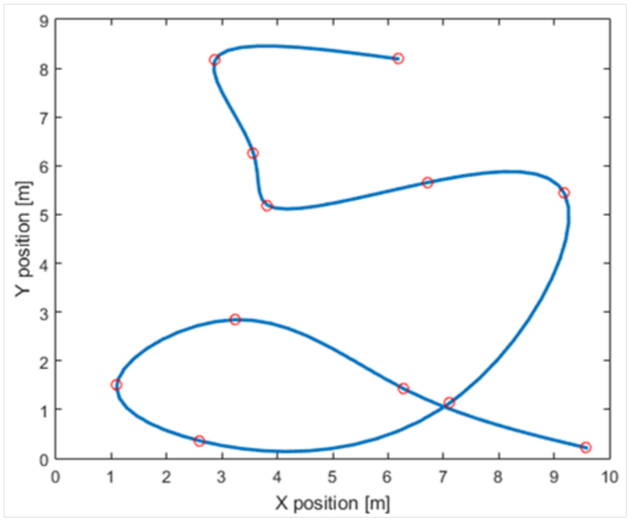

Figure 8. Solution for the trajectory optimization subsystem

3.4 Results

After several runs of the genetic algorithm converged to the same solution, the trajectory shown in Figure 8 is believed to be a global minimum. This trajectory has a minimum execution time of 17.67 seconds corresponding to a velocity of $1.90m/s$ for $a_{max} = 10m/s^2$. The optimal trajectory had a total length of $33.56m$ while the shortest trajectory found had a length of $29.35m$.

4 System Level Optimization

The two subsystems were combined to form the system-level optimization problem as seen in Table 6. The combined problem has the same objective as the trajectory subsystem, which is to minimize path time t_{path}. Instead of an arbitrary constraint a_{max}, the acceleration is now bounded by the robot's dynamics and actuator torque limitations.

4.1 Subsystem Interdependence

The formulation of the system-level optimization problem is illustrated in Figure 9. The system-level problem is composed of four interconnected analysis functions. The subsystems were combined into a nested configuration. In this configuration, an optimal robot and associated minimum path time t_{path} are calculated for a given trajectory at each evaluation of the trajectory optimizer's objective function. The Robot Optimizer determines the optimal robot configuration for a given trajectory by iteratively calling the Trajectory Analysis and Robot Dynamics functions with different robot base locations $[x, y]_c$, link lengths $L_{1,2}$ and path durations t. The Trajectory Analysis function uses the path duration to produce velocity and acceleration data for each waypoint. The Robot Dynamics function returns the torque $T_{1,2}$ and reachability constraint violations $R_{1,2}$ at each waypoint, which are passed back to the Robot Optimizer. This inner loop is iterated until an optimal robot is found for

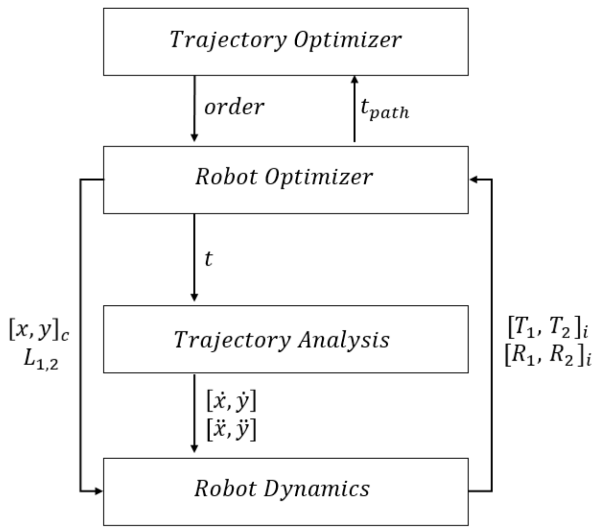

Figure 9. System level nested arrangement of subsystems

Table 6. System Level Optimization Formulation

Variables	Parameters	Objective	Constraints
p_i $i = 1..N_p$ (disc., local)	$[x_i, y_i]$ $i = 1..N_p$ (cont.)	$min\ t_{path}$	$d\lvert\mathbf{v}(t)\rvert/dt = 0$
L_1 (cont., local)	m_1 (cont.)		$p_i \neq p_j$ for $i \neq j$
L_2 (cont., local)	m_2 (cont.)		$T_{1,i} < T_{max,1}$
x_c (cont., local)	$T_{max,1}$ (cont.)		$T_{2,i} < T_{max,2}$
y_c (cont., local)	$T_{max,2}$ (cont.)		$L_1 + L_2 > max\left((x_i^2 + y_i^2)^{\frac{1}{2}}\right)$
t_{path} (cont., local)			$\lvert L_1 - L_2\rvert < min\left((x_i^2 + y_i^2)^{\frac{1}{2}}\right)$
			$L_1, L_2 > 0$

the specified waypoint visitation order. When this inner loop converges to a minimal path time t_{path}, it returns this value back to the Trajectory Optimizer, which tries to converge to a point visitation order corresponding to a global minimum t_{path}.

4.2 Problem Formulation

The system-level variables, parameters, objective and constraints are summarized in Table 6. The constraints are a combination of both subsystem constraints. They ensure that the robot can reach the trajectory, travels on the trajectory with constant speed, passes by each waypoint exactly once and respects the actuator torque limits. All of the design variables are continuous, except the waypoint order, which is discrete. They are all local design variables because they are inputs to only one analysis function. All of the parameters are continuous.

Table 7. Combined System Robot GA Optimization Parameters

Population size	75
Number of Elites	10
Generations	75

Table 8. System Analysis Function Summary

Function Name	Function Description	MATLAB Name
$\mathbf{a_1}$	Trajectory Optimizer	`Master`
$\mathbf{a_2}$	Robot Optimizer	`objFun`
$\mathbf{a_3}$	Trajectory Analysis	`accelConstraints`
$\mathbf{a_4}$	Robot Dynamics	`armConstraints`

4.3 Optimization Technique

Both trajectory and robot optimizers use genetic algorithms. The parameters for both optimizers were fine-tuned to balance solution accuracy with computational cost. The robot optimizer options for the system-level problem are shown in Table 7, while those for the trajectory optimizer are shown in Table 9. For both optimizers, the population size had to be reduced to maintain reasonable solution times. Since the robot optimizer is nested inside the trajectory optimizer, each trajectory optimizer objective function evaluation was computationally very costly. Although the robot and trajectory subsystems were interdependent, the non-linearity of the trajectory subproblem resulted in it having the dominant effect on path time. A handful of top trajectories performed significantly better than any others. Furthermore, testing indicated that the acceleration magnitude constraint used in the trajectory subproblem served as a good surrogate model of the robotic arm's dynamics constraints. With few exceptions, the path time of a trajectory using the acceleration magnitude constraints correlated well to its performance using the full system level constraints. Given this information, the system level genetic algorithm was initialized with a population of the best trajectories found using the trajectory subproblem's genetic algorithm. Due to constraints on available computational time and the high quality of the trajectory initial population provided to the system level optimizer, a relatively low population size of 8 trajectories were run for a total of 20 generations. In total, the system level trajectory optimizer performed 169 objective function evaluations. With each objective function evaluation taking an average of just under 4 minutes to perform, the total system level optimization routine took approximately 11 hours.

Table 9. System-Level Trajectory Optimizer GA Optimization Parameters

Population size	8
Number of Elites	2
Generations	20

Table 10. System level optimization results

Base position $\mathbf{x_c^*}, \mathbf{y_c^*}$ [m]	Link lengths $\mathbf{L_{1,2}^*}$ [m]	Path Time t_{path} [s]
$[8.3175, 3.54]$	$[4.08, 3.54]$	21.36

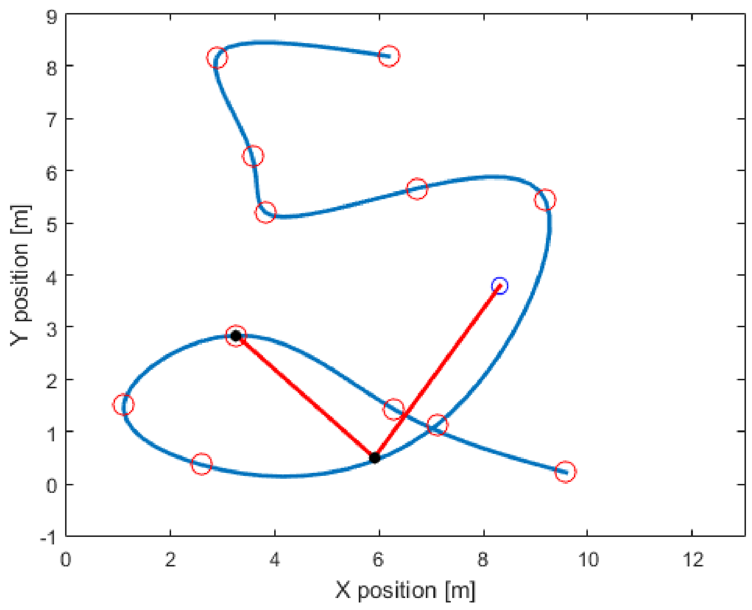

Figure 10. System Optimal Robot and Trajectory

4.4 Results

The optimal trajectory and robot found for the system level optimization problem are shown in Figure 10. The optimal robot parameters and path time value are shown in Table 10. The trajectory found to be optimal for the system level problem is the same trajectory which was found during the trajectory subsystem optimization. This reflects the fact that the acceleration magnitude constraint imposed during optimization of the trajectory subsystem was a close approximation to the dynamic constraints of the robotic manipulator. The system optimal trajectory had a path length of $33.56m$ which could be executed in 21.36 seconds. This corresponds to a constant end effector speed of $1.57m/s$. The optimal robot for the system-level problem was different than that found during subsystem optimization. The system-level optimal robot used shorter link lengths to keep its peak actuator torques below the constraint threshold. The differences between the subsystem and system-level optimal robots reflect a switch from optimizing for minimum energy usage (average torque exerted) in the subsystem to minimum path time at the system level.

4.5 Analysis

A post-optimization analysis was conducted to determine the sensitivity of the path time value t_{path} to each of the parameters and constraints. This sensitivity information can be used to inform future design efforts to decrease path execution time. Although it is possible

Table 11. Objective Function Sensitivity to Robot Parameters

Parameter	**Sensitivity** $[\Delta f\%/\Delta p\%]$
$T_{max,1}$	-0.56
$T_{max,2}$	-0.19
m_1	-0.40
m_2	0.55

Table 12. Objective Function Sensitivity to Target Positions

Target Point	**1**	**2**	**3**	**4**	**5**	**6**
x sensitivity $[\Delta f\%]$	2.78	2.62	4.98	2.35	2.20	2.99
y sensitivity $[\Delta f\%]$	-1.93	3.12	2.06	2.27	2.05	-0.45
Target Point	**7**	**8**	**9**	**10**	**11**	**12**
x sensitivity $[\Delta f\%]$	0.93	2.43	6.93	1.56	-16.38	-1.53
y sensitivity $[\Delta f\%]$	-4.15	-0.16	-1.06	1.34	1.71	2.84

that variations to the robot parameters or target points could result in a different optimal waypoint visiting sequence, repeated evaluation of the full system-level optimization problem for each parameter perturbation would be prohibitively time consuming. Instead, this parametric study assumes the waypoint visiting sequence was unchanged and examined the sensitivity of the objective function to changes in target point locations and robot parameters. The sensitivities of the objective function to each of the robot's parameters are given in Table 11. These robot parameter sensitivities are reported in the percentage change in objective function value per percent change in parameter. These sensitivities were found by re-solving the robot optimization problem using the optimal trajectory and systematically adding perturbations to one parameter at a time. Each parameter was perturbed by increasing its value by 10%. As can be seen in Table 11, decreasing m_2 or increasing $T_{max,1}$ would be the most effective means of decreasing the path time. Table 12 gives the sensitivity of the objective function to changes in the x and y position of each of the target points. The values reported in Table 12 are the percent change in the objective function value resulting from a $10cm$ perturbation of each target position in the positive x or y direction. The active constraints at the optimal solution along with their corresponding sensitivities are provided in Table 13. Actuator one's torque limit was reached at target points 7 and 9. The objective function is relatively sensitive to the positions of points 7 and 9, but is far more sensitive to the position of point 11. The fact that the objective function proved to be most sensitive to the position of a point which does not correspond to an active constraint suggests that the objective function is highly non-linear in this region. This non-linearity can also be observed in figures 11 and 12 which show cross sections of the objective function at the solution in the L_1, L_2 and x_c, y_c planes respectively. In these plots, dark blue areas represent regions where the objective function values are low while yellow areas represent infeasible solution spaces.

Table 13. Active Constraints and Sensitivities

Active Constraint	df/dh
$T_{1,7} < T_{max,1}$	-0.27
$T_{1,9} < T_{max,1}$	-0.12

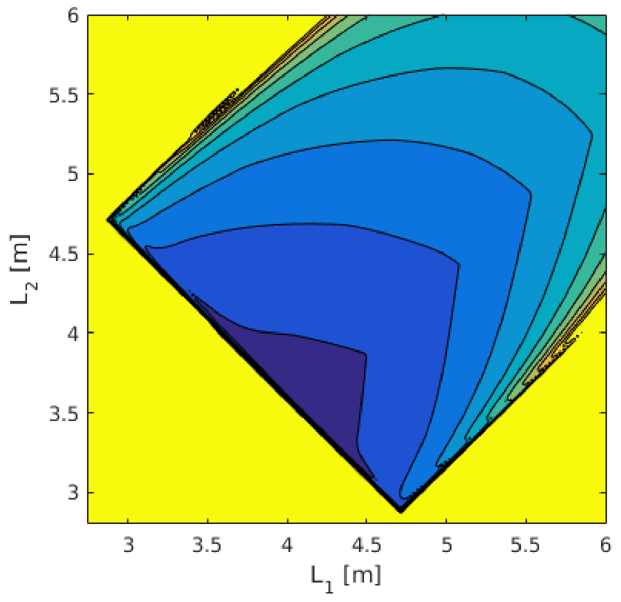

Figure 11. Objective value of link length design space at optimal base location

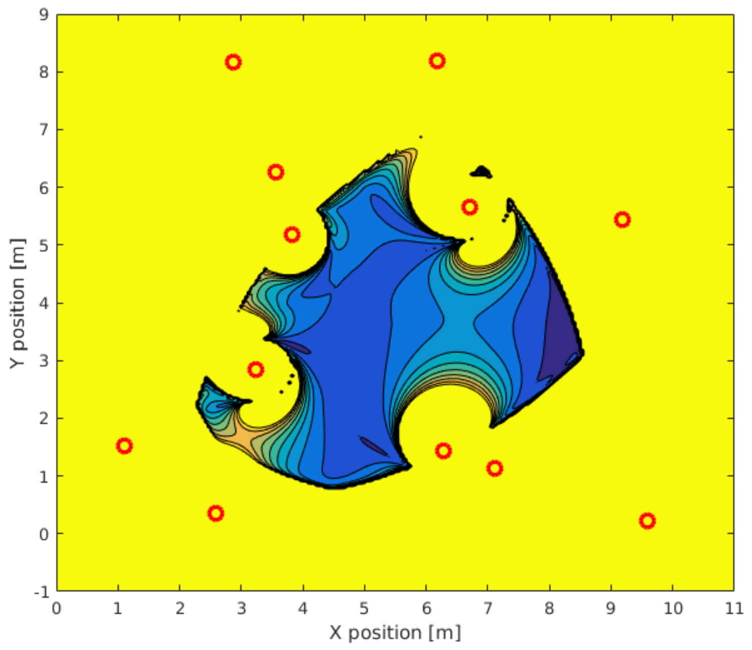

Figure 12. Objective value of base location design space at optimal link lengths

5 Future Work

Both of the subsystems were modeled under strong assumptions. Firstly, The robot model can be modified to include link inertia, length dependent link mass, joint friction, or unilateral constraints. A more realistic robot model would potentially increase the dependence of the system-level optimization solution on robot parameters. Second, the trajectory subsystem can be modified to consider varying tool tip speeds and a method more sophisticated than a cubic spline fit could be used to generate a trajectory. Moreover, the task space can include spatial constraints such as walls or walkways to simulate a more realistic environment.

6 Acknowledgments

The authors would like to thank David Remy for his input in generating a robot model, and Emrah Bayrak for his guidance throughout the project.

References

[1] D. Remy, MECHENG 543. Class Lecture, Topic:“Projection Methods.” University of Michigan, Ann Arbor, MI, Mar. 17, 2016.