Lab 1 : Optimal placement and kinematic design of a SCARA robot

MISRA Debaleena, JEANNEAU Guillaume

Ecole Centrale de Nantes : Master ARIA ROBA, OPKID

I. Presentation of the problem

The goal of this lab is to design a SCARA robot that must be able to perform any cutting trajectories in a prescribed rectangular area. The robot is constrained to operate in a workshop area of size $4m \times 4m$. For cost reasons, the link lengths should not exceed $2m$.

The placement and design is done in two phases as described in the following two sections.

II. Optimization of the base placement

Given a set of disc obstacles around the robot, determine the optimal placement of the robot base, such that the robot is able to follow:
- Any cutting trajectories in a prescribed rectangular area, i.e cover *full* rectangular area
- In case of high obstruction, cover *greatest area* possible in the rectangle

A. Robot Model

The given model is a SCARA robot (Selective Compliant Assembly Robot Arm). It has 4 axes, all parallel (vertical)- 3 revolute joints and 1 prismatic joint. The prismatic joint can be the first joint or the last joint in the chain. SCARA robots are light, very rapid, accurate and cheap. They perform *Schoenflies* motions.

Geometric modeling The equations for modeling the given robot is listed below:
Lengths : L_1=L_2= $1m$ Joints limits : $-132° < \theta_1 < 132°, -141° < \theta_2 < 141°$
The DGM and IGM equations are already provided in the lab subject, and are taken directly from there. DGM has one solution, whereas the IGM has two solutions.

B. Design of the Working Area Layout

The given workshop area is a $4mx4m$ block. A prescribed rectangular area of $1.4mx1.4m$ is selected for high obstacle definition and $1mx1m$ for low obstacle. The robot is expected to access this area. Several obstacles of varying dimensions are placed to create collision constraints that the robot must carefully avoid.

Three cases of low, medium and high obstructions are demonstrated in this work. The figure 1 show those area.

C. Methodology

A pose **X** is accessible **if and only if**: [P. Wenger Notes]

- $\exists q / X = f(q)$ (solution to the IGM)
- **q** is in the reachable joint space (within joint limits)
- $Rob(\mathbf{q}) \cap Obs = \phi$ (no collisions)

Steps

1) The entire layout is approximated by a series of equally distributed points. We have taken a resolution of 0.1 that gives us a total of $4/0.1x4/0.1 = 1600$ discrete points in the entire layout, and 196 points in the $1.4mx1.4m$ prescribed rectangular area.
2) We start from the first position P_1 at (1,1) in the layout grid. Placing the robot base at this P_1, we calculate which points in the prescribed area can be accessed. To do this:
   - The IGM for this P_1 is computed for all 196 points (x, y) of the prescribed rectangular area. If an IGM solution is found with joint angles within given limits, then it is taken as a valid solution.
   - The obstacle collision check is performed between the point to be reached by links and the robot base, using the obstacle check function (provided in the lab subject). If a valid IGM solution is reached, and the robot faces no obstacle to reach it, then this point is considered to be accessible. A counter sums up all such accessible points for P_1 among the 196 points of the rectangular area. We make separately the sum for each posture (see II-D). We keep then the best result from the two posture.
3) Step 2 is repeated for all 1600 points in the entire layout.
4) The count of total accessible points corresponding to every point in the layout, is sorted and the layout points that have the highest number of accessible points are isolated. We select one as they all are base

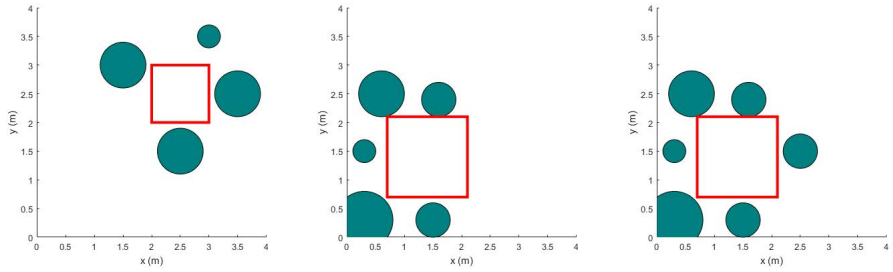

Figure 1. Working Area Layout with Low, Medium and High Obstacles

optimal placement as they give the highest area covered.

5) This procedure is inserted in a function. This function give finally the fact the area is covered or not, the number of point accessible by the best location. Moreover we use it to have the best locations and to plot their workspace if needed.

D. Discussion of Aspects and Singularity

An aspect is a separation in the available workspace. This aspects can be linked to the singularity or to obstacles and joint limits making the link between to point impossible without moving the base of the robot. The singularities of a manipulator play an important role in its global kinematic properties. However, in our task, in order to ensure simplicity, the aspect consideration due to obstacles has been omitted. The desired workspace is supposed to be connected. We just took the singularity aspect in our code. The obstacle aspect can be seen in the joint space. It could be reviewed in the future to ensure that kinematic constraints (no change of posture or aspect) are avoided. In this section, we briefly reflect on the theory of this [1].

The objective is to be able to perform cutting trajectory in the whole workspace. From this we have to be able to stay in the same configuration in the whole workspace. The singularities of a general manipulator can be found with the determinant of the Jacobian matrix $det(\mathbf{J})$. The contour plot of $det(\mathbf{J}) = 0$ divides the joint workspace into singularity-free domains. The **aspects** are the largest singularity-free connected regions of the joint space. They are bounded by the singularity surfaces and by the joint limits when they exist. Additional boundaries may occur in the presence of obstacles. Before cuspidal robots were discovered, one posture was thought to be uniquely associated with one aspect. Indeed, since the robot was thought to cross necessarily a singularity when moving from one posture to another, there could not be more than one posture in each of its aspects. This is true for most usual industrial robots like the anthropomorphic robot and the SCARA robot, as in our task.

For the SCARA model, the determinant of the Jacobian matrix $det(\mathbf{J}) = 0$ when $sin(\theta_2) = 0$, giving rise to a singularity whenever $\theta_2 = 0$ or $\theta_2 = \pi$, corresponding to fully extended or fully folded configurations. So, this robot has two aspects defined by $\theta_2 > 0$ and $\theta_2 < 0$. Also, the IGM gives two solutions - linked with postures of elbow up or elbow down. And it is found that there is only one IGM solution in each aspect, hence a posture is associated with one aspect. The figure 2 shows on the top the two aspects in (θ_1, θ_2), and on the bottom, shows the posture in workspace. It show that only an elbow-down posture can access the workspace.

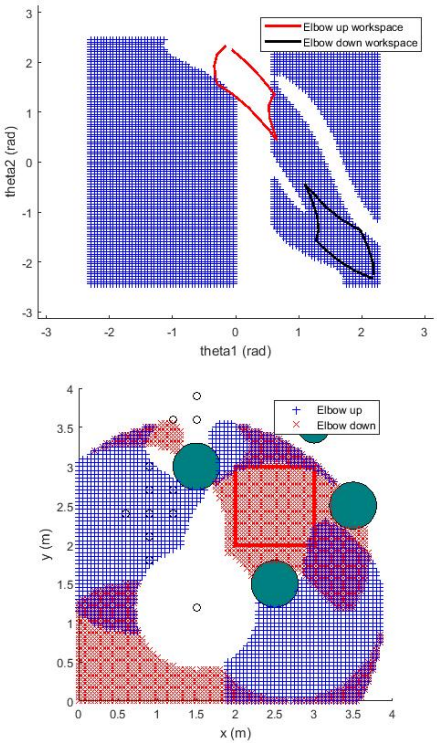

Figure 2. Aspects of the SCARA robot in the joint space (left) and workspace (right)

We can see in the figure 2 that there is two unlinked space. If we report this workspace in

the effective workspace, we can define the real available workspace. Those aspect are separated from aspect linked with singularity, this time, the two area in the joint space will never be joined even if we cross singularity.

III. Optimization of the link length

The objective is to minimize the link length of the scara robot. The problem is we want the robot to always reach the all define workspace. It may be impossible. In this case, we want to optimize the covered area of the workspace. The link length as much as possible the smallest possible.

Several possibilities where available for this optimization. We first had to choose the variable to optimize. Then we have to choose a proper optimization function we want to minimize. We may have moreover some constraint.

A. Variable to minimize

We have at this point five variable to optimize. The link lengths (L_1,L_2) and the robot initial pose (x_0,y_0,ϕ). The angle ϕ define the orientation of the base. It is an optimization variable because the base orientation define the orientation of the joint limit. We used a change of frame to apply the IGM and the DGM and then now exactly how the robot behave.

We already did an optimization previously. The previous optimization gave the optimal placement on a discrete grid leading to best area coverage of the workspace. At first, chose to use this optimization because it give all the best position and don't focus on local maxima. This way we effectively look on all the workspace. We could miss some very local minimum, but the color map of the number of point reached by the robot for each point seam to show that this is a continuous function with a variation step close to the step we choose. We think then that we don't miss any particular point. To improve the accuracy, it is possible to increase the accuracy around the best solution. The drawback of this solution is that it lead to a lot of computation. The discretization step must be quite high, leading to a very bad accuracy. That is why we finally decided to use the optimization of all 5 parameter. The optimization became then much more quicker.

We finally assume that variable to optimize are then the link length and the base pose and angle. The optimization of the workspace can be down in the optimization function or in the constraint. We chose to make it in the optimization function.

B. Objective function

The objective function f is the function we want to minimize. It take in argument the optimization variable and return a scalar to minimize. It takes then the link length and base orientation in argument.

There is two different cases. The link length can lead to a total coverage of the workspace or not. This is verified by a function $CheckWorkspace(L_1, L_2, phi, x_0, y_0, robot, workspace, area)$. This function will return first a boolean. If there is a total coverage of the workspace with one posture, it will be equal to 1, else it will be 0. The number of point N_{point} reachable by the workspace is the other output. This function is explained on part II-C.

1) Area totally covered: The first objective of this function is to minimize the link length. We will apply this minimization if the workspace is totally covered. To do that, we choose to minimize the sum of the link length $L = L_1 + L_2$.

2) Area not totally covered: If there is no total coverage, we will maximize the number of point reached by the robot. As we do a minimization, we will minimize its inverse. There is two problem with this optimization. The number of point reached is a constant piecewise function of the link length and we want preferably to minimize the link length.

a) Number of point is a constant piecewise function of the link length: The first situation is a problem because an optimization algorithm like the one available $fmincon$ will consider a constant function as a local minimum for every position. To overcome to the first problem, several possibilities are possible. The first option is to use a solver for the optimization working for discrete function like $simulannealbnd$. This can be done by a simulated annealing method. A very good other solution could be too to use genetic algorithm. As we didn't have access to this solver, we didn't use it. At the beginning, we used the function $fmincon$. This function is not suitable for the maximization of the workspace when the workspace is not totally covered. It can thus optimize the link length when it is possible. That is why we finally change to the $simulannealbnd$ function.

b) Prioritizing the total coverage of the workspace: The second problem is that we want the first criteria of total workspace reached to be the priority one. Indeed, it could be possible that the value of the function f lead to a lower output with a non totally covered workspace than a covered workspace.

To avoid this situation, we chose to add a constant in the case the workspace is not totally covered. This constant is chosen to be the maximum of the sum of the link length, meaning $4m$.

3) Objective function implementation: Finally the function to minimize is the following one :

4) Constraint: Finally the constraint such as the obstacles and the joint limits have been included in the optimization function. The only constraint

Data: $L_1, L_2, phi, x0, y0$
Result: $output$ to minimize
$[N_{point}, bool] =$
$CheckWorkspace(L_1, L_2, phi,$
$x_0, y_0, workspace, area)$
if *bool* **then**
  $output = L_1 + L_2$;
else
  $output = 4 + 1/N_{point}$;
end

Algorithm 1: Objective function

remaining are the maximum and minimum value of the optimization variable.

IV. Results

A color map is presented for comparison of which areas in the layout has the maximum coverage of area in the prescribed rectangle- *yellow* end of the spectrum having most area that drops as we move to the *blue* regions. We plot to the workspace of one of the best point to see what it look like. With *blue* cross we plot one posture workspace (elbow up), with *red* cross we represent the other one (elbow down). On figure 3,4,5, we can see the comparison between the pre and post optimization in presence of low, medium and high obstacles. The joint-space figures are given at the end of the report, for additional insights.

A. Low obstacles result

The low obstacle first cases shown on figure 3 has been chosen because a lot of different case could be chosen all around the workspace. The problem is to chose which one could be the best pose. Without optimization of the angle we see that no point appear suitable close to the final solution on the first graph. After optimization, this spot appears and give good result. The link length have been minimized, going from $l_1 = 1m$ and $l_2 = 1m$ to $l_1 = 0.73m$ and $l_2 = 1.25m$. We see that the link length sum was close to the best solution. The advantage of the colormap is that it give a good idea of what could be the good base position.

B. Medium obstacles result

The figure 4 show here that it work too when the link length are too short. After optimization, the link length are equal to $l_1 = 1.36m$ and $l_2 = 1.35m$. The algorithm optimize the number of point to reach and we finally get the coverage of the workspace. We see here that a lot of pose are possible for the robot. The program is not the optimal one because it is quite easy to find shorter link length. As the program look quite randomly a solution, this program is good to find effective solution but not the best one. This algorithm is very sensible to initial condition at the beginning of the optimization. It could be good to use a random input to see how well it is possible to improve the solution. Other optimization tool like genetic algorithm could be used to.

C. High obstacles result

For the high obstacle case, we see that the program succeed to find a solution where all coverage with only one configuration is achieved for the algorithm. However if we look in the joint space or if we look with a better resolution, we see that the workspace is not completely covered. The resolution could then be changed to find better results. We are moreover close to singularity, It could be good then to look for improving the dexterity of the robot.

V. Dexterity improvement

Once the optimization have been made, we can try to improve the dexterity of the robot. The first step is to calculate the dexterity of the robot. Then we will try to optimize it.

A. Calculation of the dexterity

The dexterity is the ability for a robot to work on every direction. It can be calculated using the manipulability index. This one is define for a $(n \times m)$ matrix $\mathbf{M}$ as :

$$\kappa(M) = \frac{1}{m}\sqrt{tr(\mathbf{M}^T\mathbf{M})tr((\mathbf{M}^T\mathbf{M})^{-1})} \quad (1)$$

This definition is applied on the Jacobian matrix of the robot. On our case, the Jacobian matrix is equal to:

$$\mathbf{J} = \begin{pmatrix} -L_1\sin(\theta_1) - L_2\sin(\theta_1+\theta_2) & -L_2\sin(\theta_1+\theta_2) \\ L_1\cos(\theta_1) + L_2\cos(\theta_1+\theta_2) & L_2\cos(\theta_1+\theta_2) \end{pmatrix} \quad (2)$$

This index must be as small as possible. When this index is equal to 1, it mean that this position is isotropic. The robot will behave the same way in all the directions. When the index approaches to infinity, it means that the position is singular, one degree of freedom is lost.

B. Improvement of the algorithm for dexterity improvement

This index is not constant within the workspace. On one position it can tend to infinity, and on an other one, it can tend to one. We have then to define properly the objective. We guess that we want no singularity in the workspace and that we want to have an area where the dexterity is really improved.

1) No singularity in the workspace: To avoid any serial singularity in the workspace, we can bound the manipulability index. In a constraint function, we can calculate the manipulability index in all the workspace and verify that the maximum value is lesser than a given constant. The solution, if it exist, will then avoid singularity.

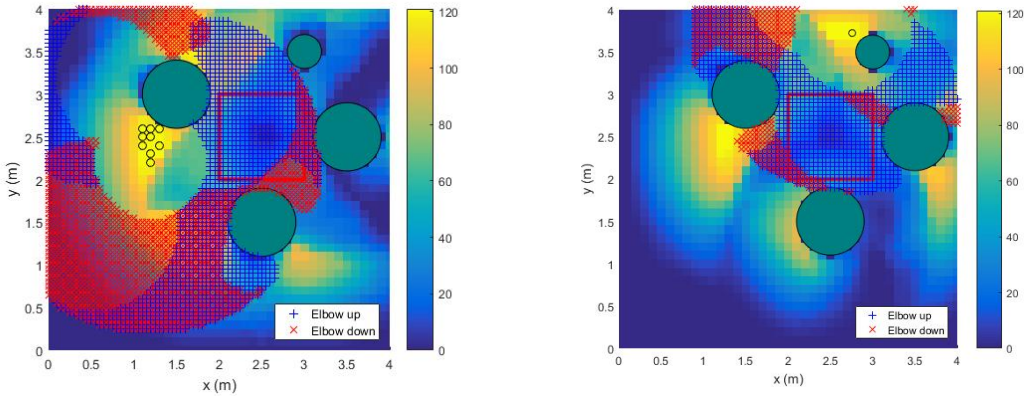

Figure 3. Workspace reached and number of point reached for each point of the workspace before and after optimization with low obstacle

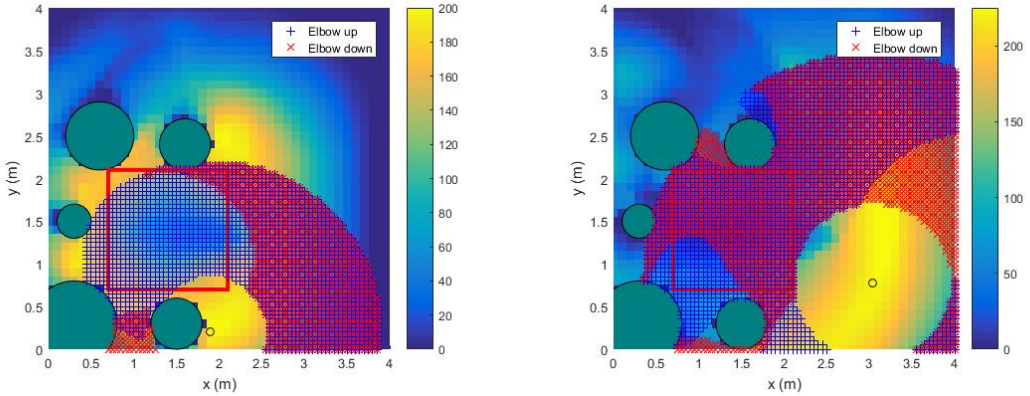

Figure 4. Workspace reached and number of point reached for each point of the workspace before and after optimization with medium obstacle

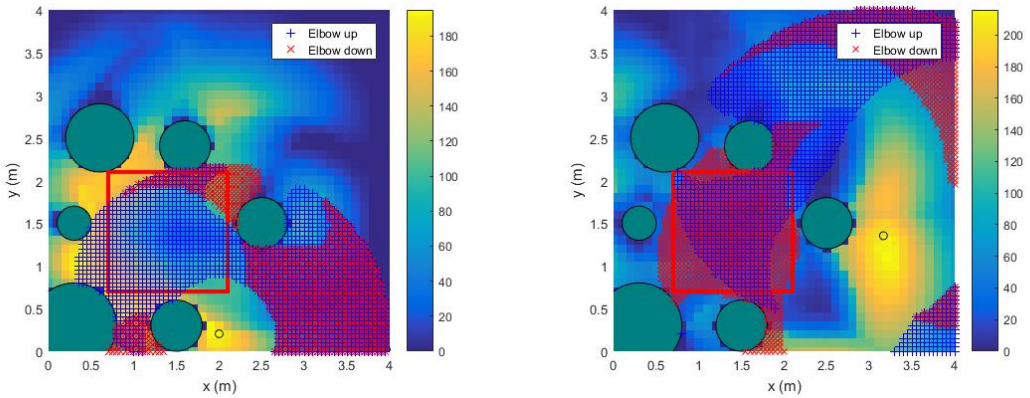

Figure 5. Workspace reached and number of point reached for each point of the workspace before and after optimization with high obstacle

2) Dexterity improved on a point: To improve the dexterity, it is possible to minimize it on one point. We can minimize the minimum dexterity in the workspace. This way, the robot will try to find an isotropic position for itself.

C. Conclusion

This work presents an investigation of optimal kinematic parameters of the SCARA robot in application to accessing a given workspace. The main objective is to maximize the reach of our robot, taking into account the obstacle avoidance. We have ensured that both geometric and kinematic constraints are considered while optimizing the link lengths and base placement. It has been seen that our algorithm runs successfully and gives higher coverage of workspace, with a minimized link lengths. Several suggestions have been included in the discussion of dexterity, that maybe used in a future extension of the current work.

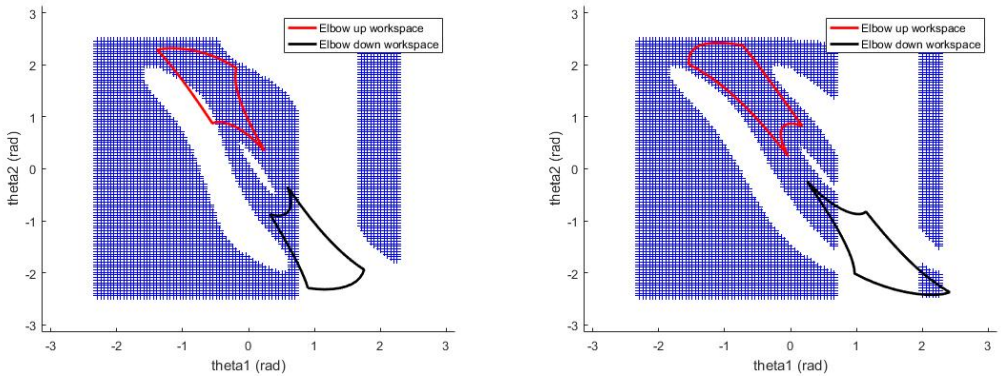

Figure 6. Image of the desired workspace in the joint space and workspace of the robot with low obstacle before and after optimization

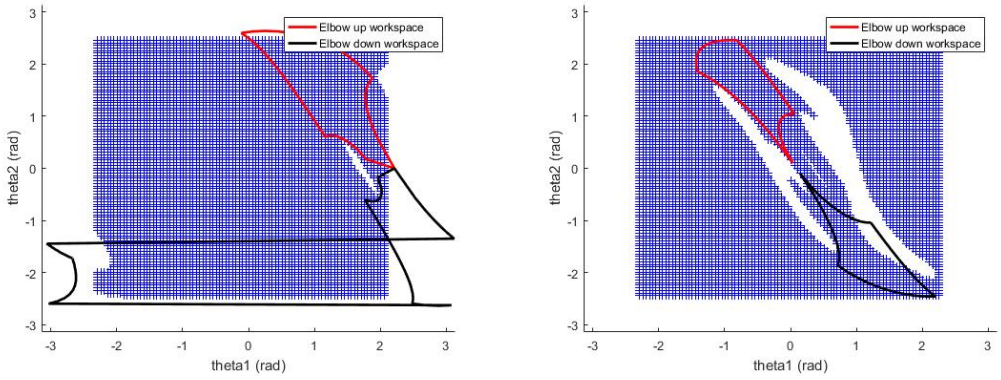

Figure 7. Image of the desired workspace in the joint space and workspace of the robot with medium obstacle before and after optimization

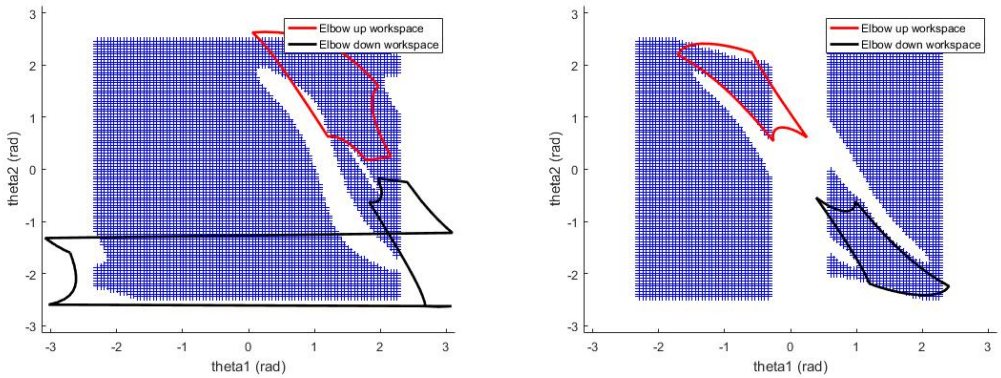

Figure 8. Image of the desired workspace in the joint space and workspace of the robot with high obstacle before and after optimization