86739: Optimal Kinematic Design of Robots
Optimal placement and kinematic design of a SCARA robot

Date: Tue 23:59, 01.11.16

Prof. P. Wenger

Rabbia Asghar, BEng, Ernest Skrzypczyk, BSc

Chapter 1

Optimal placement and kinematic design of a SCARA robot

Experimental setup

The problem statement incorporates optimal placement and kinematic design of a SCARA (Selective Compliance Assembly Robot Arm or Selective Compliance Articulated Robot Arm) robot using specified restrictions in order to perform cutting trajectories.

The SCARA robot consists of 4 parallel axes in the vertical plane with 1 prismatic and 3 revolute joints, which enable Schoenflies motions, ergo 3 translations and 1 rotation about a vertical axis. The experimental setup consists of a 2 bar with initial lengths set as $l_1 = l_2 = 1\ m$. The robot is subject to several constraint presented in Table 1.1. The simplified SCARA version in the limiting configurations is shown in figure (1.1).

Table 1.1: Parametrization of simplified SCARA robot

Parameter/Constraint	Lower limit	Upper limit
Joint θ_1	-132 [°]	132 [°]
Joint θ_2	-141 [°]	141 [°]
Link lengths $l_1 + l_2$	0 [m]	2 [m]

Workshop definition

The workshop consists of a rectangular area of size $4\ m \prod 4\ m$. Three different types of scenarios have been studied with the developed optimization algorithm: low, medium and high obstruction as illustrated in Figure 1.3.

Optimal placement of the base algorithm

In order to follow a trajectory designed for the cutting process, the robot's end effector must be inside t-connected regions at all time, ergo reachable workspace is within the same working mode of the robot for all reachable regions. The general approach for the placement of the robot is represented 1. The algorithm shows the brute-force search method, where every point in the workshop area is analysed. This method was only used to find optimal placement of the robot base with fixed limb lengths and specified disc obstacles. It was conducted in order to evaluate and confirm that all functions running properly. Figure (1.5) shows

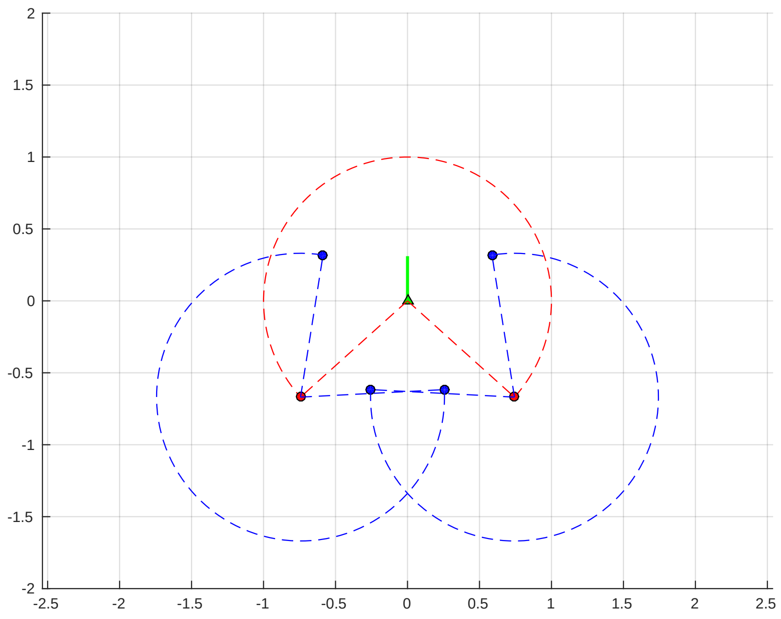


Figure 1.1: Simplified SCARA robot represented as two bars in the limiting configurations.

🟥 – link 1, 🟦 – link 2, 🟩 – base with initial orientation

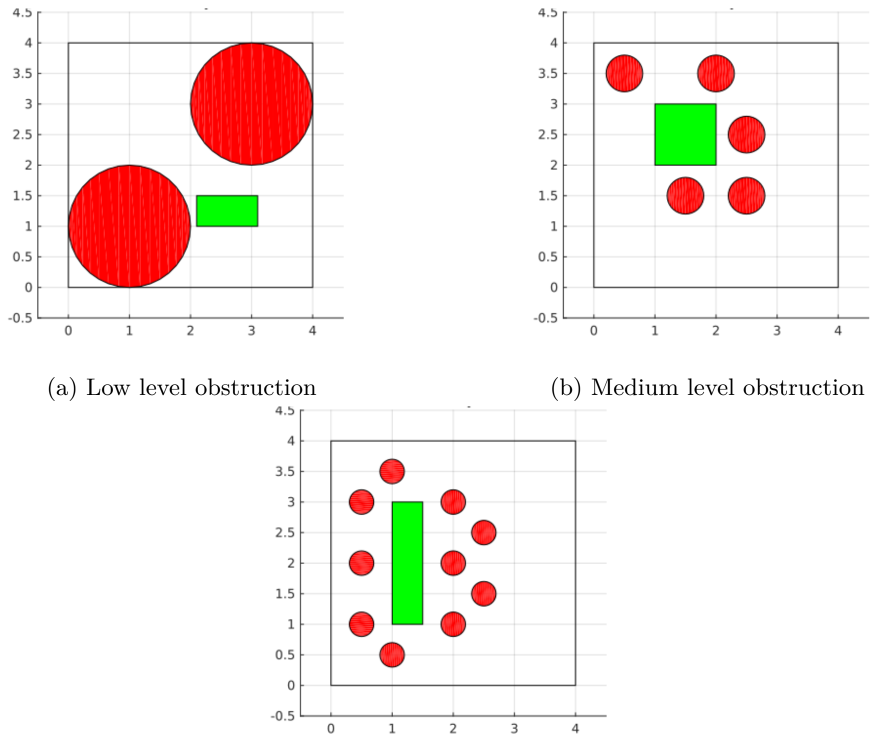


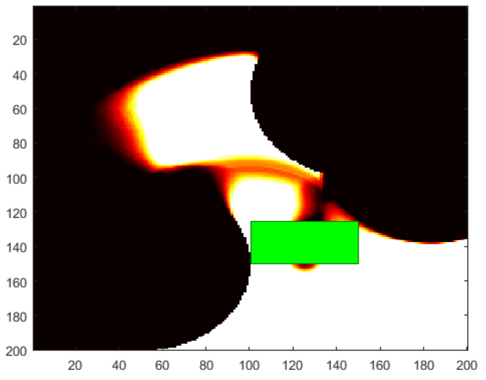
Figure 1.3: Workspace definitions with different obstruction levels.

🟩 – desired trajectory area, 🟥 – obstacles results for optimal robot base placement within the workshop 1 scenario with low level obstruction and relatively small desired trajectory area.

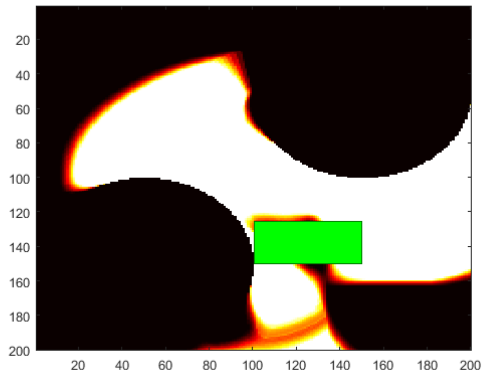
The Matlab function simulannealbnd is used afterwards to find the optimal placement of the robot with similar algorithm. Figure (1.5a) to (1.5c) shows t-connected regions for the best robot placement proposed by optimization algorithm in 3 different workshop scenarios for fixed limb lengths.

Algorithm 1 Optimal placement of robot base (brute-force search)

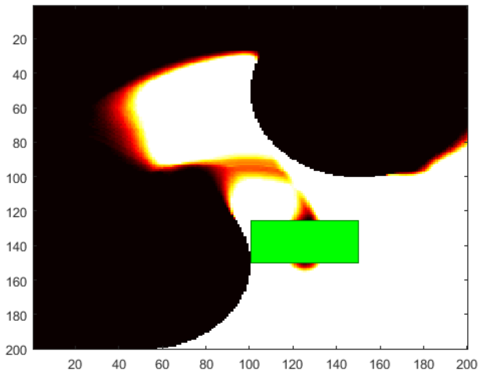
```
Require: find optimal robot placement
Ensure: cover maximum area in the rectangle
  discretized all points inside workshop
  for each point in workshop do
    identify free space
    if point is inside prescribed rectangle then
      it is not free space
    else if point is inside any obstacle then
      it is not free space
    else
      point is free space
    end if
  end for
  for each point in free space do
    for each point in the prescribed rectangle do
      compute $\theta_1$ and $\theta_2$ for both aspects using IGM
      if robot does not collide with any obstacles in Aspect 1 then
        if $\theta_1$ and $\theta_2$ in Aspect 1 are within joint limits then
          increment the number of points covered in prescribed rectangle for current point
        end if
        if $\theta_1$ and $\theta_2$ in Aspect 1 are within joint limits then
          increment the number of points covered in prescribed rectangle for current point for the case of
          varied base orientation
        end if
      end if
      if robot does not collide with any obstacles in Aspect 2 then
        if $\theta_1$ and $\theta_2$ in Aspect 2 are within joint limits then
          increment the number of points covered in prescribed rectangle for current point
        end if
        if $\theta_1$ and $\theta_2$ in Aspect 2 are within joint limits then
          increment the number of points covered in prescribed rectangle for current point for the case of
          varied base orientation
        end if
      end if
    end for
  end for
  determine point[s] with maximum number of points covered in prescribed rectangle
```



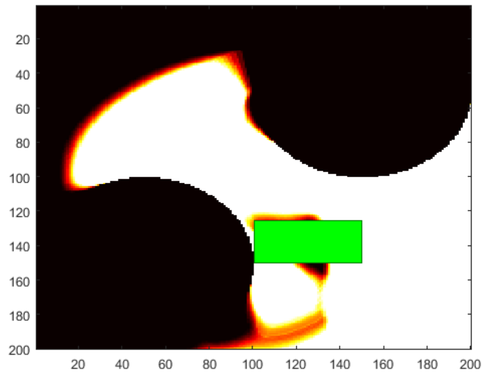
(a) Robot aspect 1



(b) Robot aspect 2



(c) Robot aspect 1 with base rotation



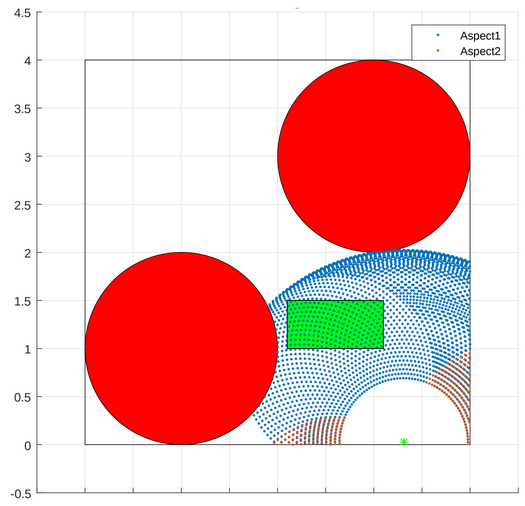
(d) Robot aspect 2 with base rotation

Figure 1.5: Optimal placement results from the brute-force search algorithm for workspace 1 scenario with low level obstruction.

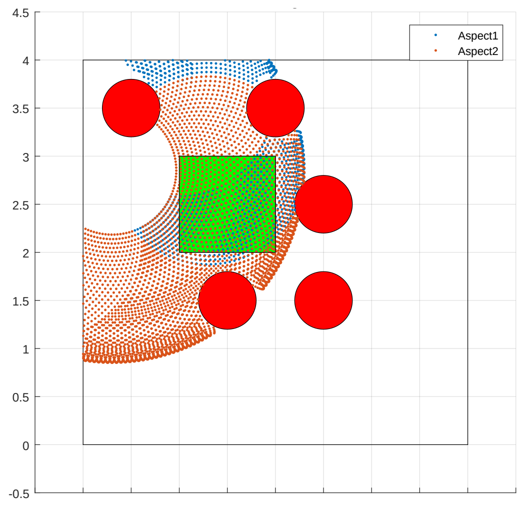
🟩 – desired trajectory area, ■ – obstacles and placements without solutions

Optimisation of link lengths algorithm

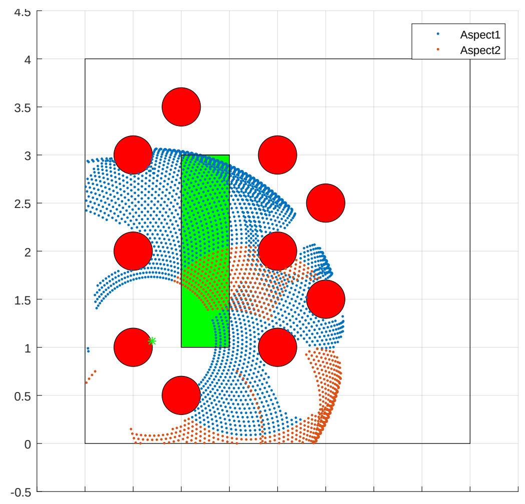
For the optimisation of link lengths, only the Matlab function simulannealbnd was used. In the objective function for this optimisation, 4 input arguments were defined: x and y coordinates of robot base and length of both limbs. Similar algorithm was used as before with slight changes. The main difference are the limb lengths l_1 and l_2 as input arguments. Figure (1.7d) represents t-connected regions for the optimal robot placement and limb lengths proposed by algorithm in third workshop scenario with high level obstruction.



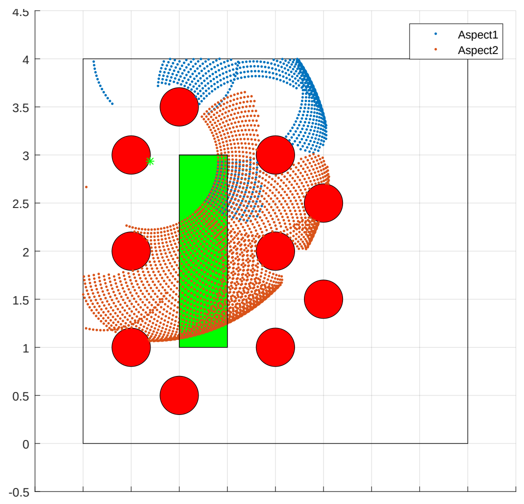
(a) Workshop scenario 1, fixed limb lengths $l_1 = l_2 = 1\ m$, optimal robot base position $x = 3.310800, y = 0.023900$



(b) Workshop scenario 2, fixed limb lengths $l_1 = l_2 = 1\ m$, optimal robot base position $x = 0.301300, y = 2.851700$



(c) Workshop scenario 3, fixed limb lengths $l_1 = l_2 = 1\ m$, optimal robot base position $x = 0.693600, y = 1.064900$



(d) Workshop scenario 3, optimal limb lengths $l_1 = 1.117400\ m, l_2 = 0.754300\ m$, optimal robot base position $x = 1.117400, y = 0.754300$

Figure 1.7: Optimisation results for different workshop scenarios with and without limb length optimisation.

Dexterity of the robot

Dexterity describes the robot's end effector ability to efficiently perform small and arbitrary displacements at a given configuration. It is related to operational velocities and joint rates as well as singularities. Dexterity is related to the condition number κ of the Jacobian matrix J defined as the relation between the highest and lowest value of Jacobian as shown in (1.1).

$$\kappa = \frac{\sigma_{max}}{\sigma_{min}} \tag{1.1}$$

The relation (1.1) needs to relate values with same units or use relative values in reference to a nominal. In order to ensure optimal dexterity, the condition number κ should be also optimal, which in practice means an isotropic configuration $\kappa = \kappa^{-1} = 1$, ergo the velocity and force relation is equal in all directions at a given configuration.

Since dexterity depends on the Jacobian of the robot, it will vary with different limb lengths and at different angle. Thus, it will vary with every iteration. In order to also optimize the dexterity of the robot κ^{-1}, a threshold can be assigned. If the dexterity exceeds that limit, zero for rectangle area to reject that case.

Another possible approach is to assign a certain weight to dexterity in the output of the objective function, for example 5 to 10%. This way the area of rectangle will have a weight of dexterity and the function will optimise the link lengths and robot placement accordingly.

Conclusions

As expected, the brute force method was far more computationally expensive than using the optimisation algorithm. However, it shows a very extensive evaluation of the workshop area highlighting the area covered in the rectangle for all the points. This method was only used for workshop scenario 1 with low level obstruction. Using optimisation algorithm, obtained the best placement of the robot in very little time. For workshop scenario 1 and scenario 2 with medium obstruction, the algorithm was easily able to find a point that covered all area in prescribed rectangles with fixed link lengths and ended after 1000 iterations. On the other hand, workshop scenario 3 was more complex with high level of obstruction, where the prescribed rectangle was surrounded by 10 obstacles. For fixed link lengths the algorithm could only cover about 70 % at best of the desired rectangle area. When the same scenario was tested with ability to optimise the limb lengths, about 75 % of the rectangle area was covered in maximum. So the specific scenario is highly restricting an optimal result by 25 % with a gain of merely 5 % using the optimized limb lengths algorithm over the fixed limb approach.