FDS 2024: Field and service robotics
Homework 1

Andrea Morghen

Question 1: Consider the ATLAS robot from Boston Dynamics. If ATLAS actuators can produce unbounded torques, establish whether each of the following statements is true or not, and briefly justify your answer.

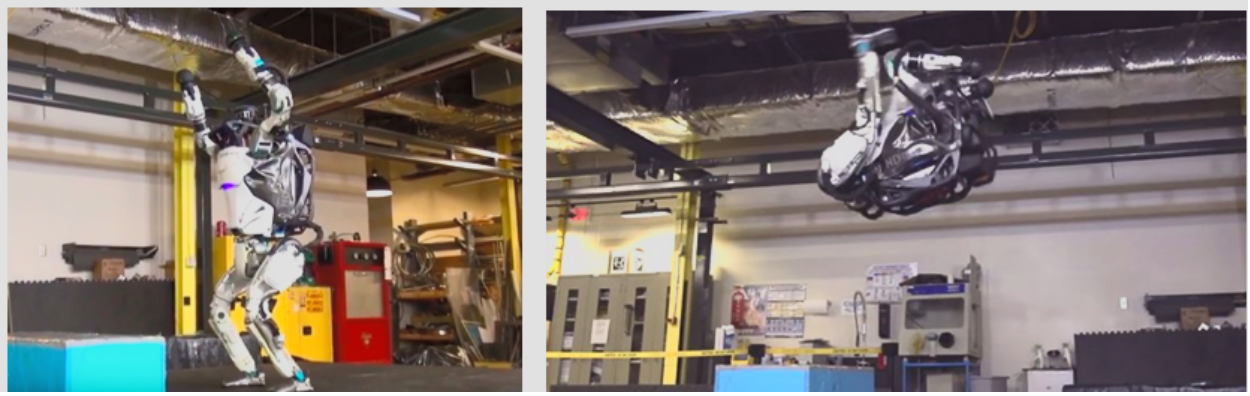

Figure 1: Atlas Robot

Part (a): While standing, ATLAS is fully actuated.

The Atlas robot is a bipedal robot; it takes the human body as its inspiration. As stated on the course slides[2]: "The human body has an incredible number of actuators (muscles), and in many cases has multiple muscles per joint;..." it is therefore possible to assume that similar to the human body, Atlas has a large number of actuators, which make it capable, in the condition of contact with the ground, of performing an instantaneous acceleration in an arbitrary direction in q, where q are the variables of the model (in this case position and orientation of the robot itself). Indeed, we recall that[2]: "... we will refer to underactuation as a property of the mathematical model (q) used to model our robotic system." The ground contact condition is crucial for full actuation.
Therefore, Atlas is fully actuated while standing.

Part (b): While doing backflip, ATLAS is fully actuated.

As stated on the slides[2]: "[The human body] despite having more actuators than position variables, when we jump into the air there is no combination of muscle inputs that can change the ballistic trajectory of the centre of mass (apart from aerodynamic effects)." The same can be said for the Atlas robot. Underactuation and full actuation are closely related to the current state, time and constraints. In this case, the absence of the ground contact constraint makes the robot underactuated in this condition. Recall, however, that to define an underactuated system (course slide[2]): "a system is underactuated if it is underactuated in all states and times". So Atlas is not an underactuated system, but it is underactuated while doing a backflip.
Therefore, Atlas is not fully actuated while doing backflip.

Question 2: Consider the planar mechanism on the left and the spatial mechanism on the right. Determine the number of the degrees of freedom for each mechanism, comparing the result with your intuition about the possible motions of these mechanisms. For each mechanism, write its configuration space topology.

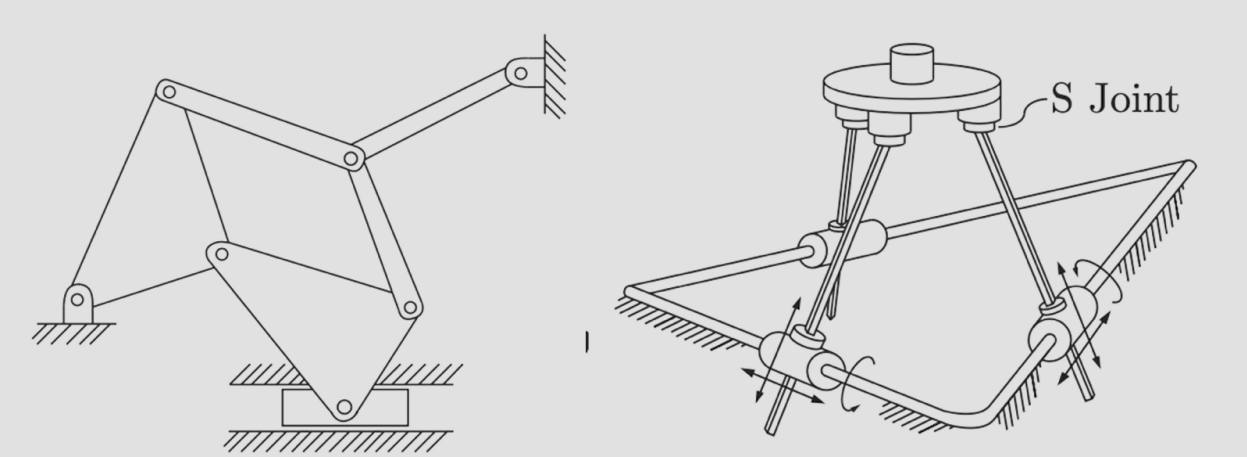

Figure 2: Mechanisms

To calculate the number of Dofs of the two mechanisms, Grübler's formula can be used

$$DoFs = m(N - 1 - J) + \sum(f_i)$$

For the the planar mechanism applies:

$m = 3$
$N = 7$
$J = 9$

8 Revolute joints ($f_i = 1$), 1 prismatic($f_i = 1$).
So the Grübler's formula:

$$DoFs = 3(7 - 1 - 9) + 9 * (1) = 0$$

According to intuition, the mechanism seemed to possess 1 degree of freedom, but in reality it is a structure (0 Dofs).

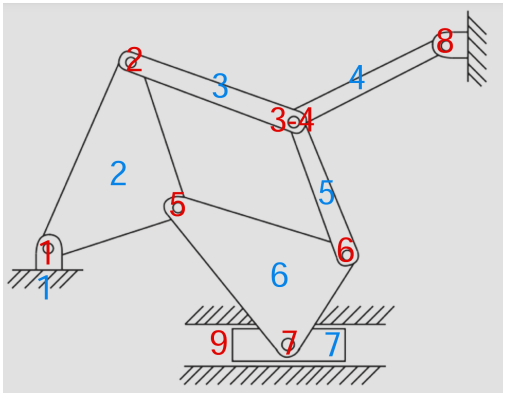

Figure 3: Planar mechanism

The configuration space topology
Being a structure, the topological space is null, no degrees of freedom.

For the the spatial mechanism applies:

$m = 6$
$N = 8$
$J = 9$

3 prismatic joints ($f_i = 1$), 3 Cylindrical($f_i = 2$) 3 Spherical ($f_i = 3$).
So the Grübler's formula:

$$DoFs = 6(8 - 1 - 9) + 3 * (1) + 3 * (2) + 3 * (3) = 6$$

According to intuition, the mechanism would appear to have 6 degrees of freedom (complete freedom of the end-effector/plate in space), as confirmed by the formula.

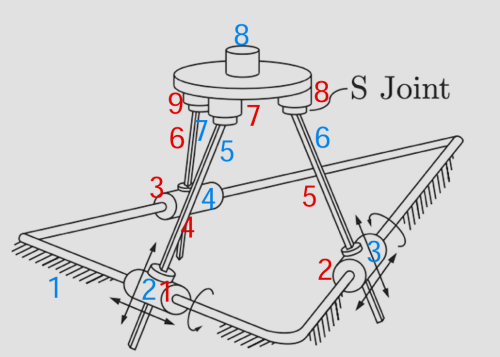

Figure 4: Spatial mechanism

The configuration space topology

$$R^3 * T^3$$

Since the mechanism has 6 Dofs, the topology space will have the same dimensions. You can choose n = 6 variables to describe this space, these variables are some of the variable joints of the mechanism. The chosen joints (actuated joints) are considered independent joints, while the others joints are the dependent ones (passive joints). $R^3 * T^3$ are the space of the 3 Cylindrical joints and have been chosen to describe the topology space of the mechanism.

Question 3: State whether the following sentences regarding underactuation or fully actuation are true or false. Briefly justify your answers.

Part (a): A car with inputs the steering angle and the throttle is underactuated.

Yes. The car cannot generate any acceleration in the space (cannot generate a lateral acceleration). From the slides[2]: "A car is underactuated because it has two control inputs (steering and forward/backward speed) but at least three DoFs given by the planar rigid body chassis".

Part (b): The KUKA youBot system on the slides is fully actuated.

Yes. The KUKA youBot is the combination of a wheeled robot with four independent actuators for each of the wheels and a 5Dofs Robotic arm. If we consider the positional and orientation variables of the end-effector, as the model variables q of interest, we can state that it can generate any acceleration in an arbitrary direction in q (considering the real joints max actuation torque).

Part (c): The hexarotor system with co-planar propellers is fully actuated.

No. The hexarotor system with co-planar propellers cannot generate any instantaneously acceleration in the space (cannot generate a lateral acceleration / orizontal plane).

Part (d): The 7-DoF KUKA iiwa robot is redundant.

Yes. The 7-DoF KUKA iiwa robot is intrinsically redundant, it has 7 DoFs in a 6 dimensional space, so intrinsically it has 1 Dof exceeding (according to the task it can have more than 1 Dofs exceeding, functional redundancy).

Question 4: RRT method

The matlab script contains the solution with a brief explanation of the code. The file is on the github repository: https://github.com/Andremorgh/FSR2024.git [1]
The file is in the folder HW1 and it is named **HW1_es4.mlx.**
A high-level abstraction explanation of the implemented programming logic is given here. Run the matlab program directly to view the full operation of the code.

Single-tree version of the algorithm was implemented, (there is a double-tree algorithm, Bidirectional RRT). To run this code the **Statistics and Machine Learning Toolbox** of Matlab is required. (the function knnsearch is a very usefull function since it deals with the identification of the nearest point in a single line! However this task could have been implemented with different lines of code without this function).

After importing the matrix map and defining some key variables, such as qf and qi, there is a while loop to implement the algorithm. The conditions of the while loop are such that the loop stops if the path is created correctly or if the number of iterations exceeds the maximum value. The main steps of the loop are as follows:

1. a random qrand point is extracted from the map
2. the closest point (qnear) to qrand is extracted from the list of points
3. the line in the direction of qrand of length delta starting from qnear is calculated
4. with a simple collision detection algorithm, the validity of all points belonging to the line is checked
5. If the collision detection check is successful the end point of the line (qnew) is added to the list of points
6. every tot iterations the possibility of connecting qf to the tree is checked, the first policy chosen for inserting qf in the tree is that the minimum distance of qf from one of the tree points is less than delta. If this condition is verified an attempt is made to insert qf into the list of points (checking for collisions on the join line), if the insertion is successful the while loop is aborted.

During the while loop the various points added to the list are printed on the screen on the map plot with the following result (delta = 20):

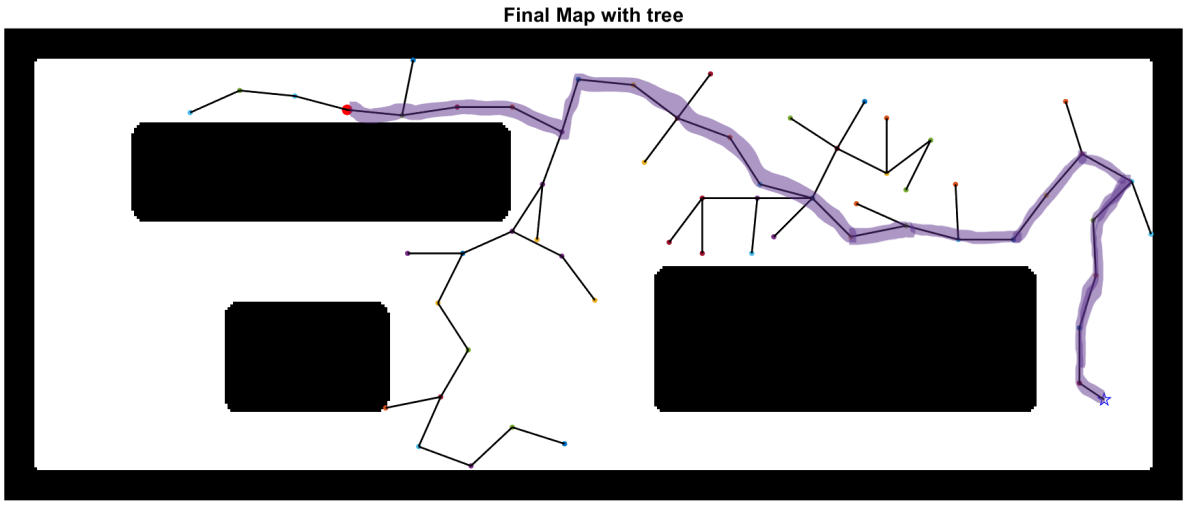

Figure 5: Final Map delta = 20

If the algorithm is unsuccessful in the maximum number of iterations chosen, it is possible to increment the n_exec parameter and repeat the execution. It is also possible to vary the delta parameter to change the step length. Shown below are two executions, respectively with delta = 15 and delta = 40. It can be observed that for lower deltas, the number of iterations required to complete the algorithm increases:

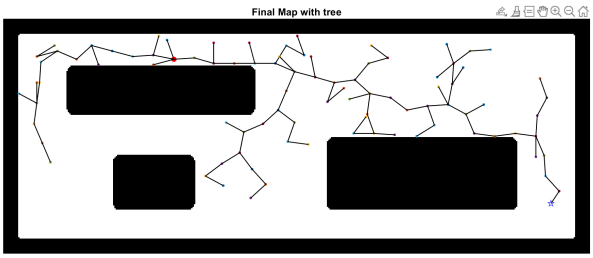

Figure 6: Final Map delta = 15

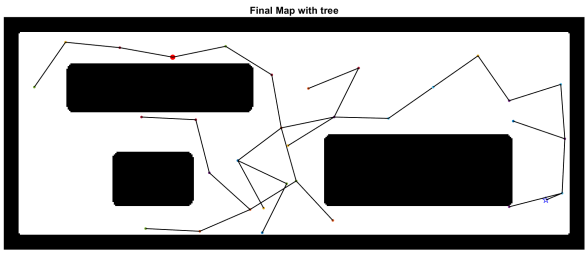

Figure 7: Final Map delta = 40

Question 5: Numerical navigation method

The matlab script contains the solution with a brief explanation of the code. The file is on the github repository:
https://github.com/Andremorgh/FSR2024.git [1]
The file is in the folder HW1 and it is named **HW1_es5.mlx.**
A high-level abstraction explanation of the implemented programming logic is given here. Run the matlab program directly to view the full operation of the code.

First the map is defined as a matrix of 0 (obstacles), 1 (clear road), 2 (goal), 3 (initial position), then some supporting variables for execution are also defined. The program is structured of two main phases:

1. **Numbered map generation**, starting from the box with numerical value i, the value i+1 is placed in the adjacent boxes (4 or 8 depending on the choice) that do not contain obstacles; the process is repeated until the map is completed or until the maximum number of iterations is reached (safety control)
2. **Path calculation**, from start point to end point according to the optimal path (at each step we go to a lower numerical value box)

Once these steps are completed, the complete path map is plotted:

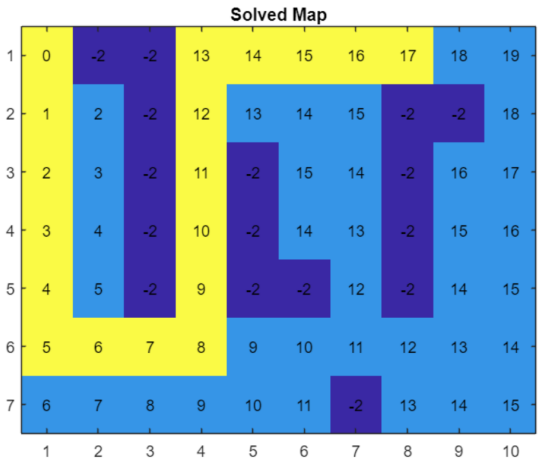

Figure 8: 4 adjacent square solution

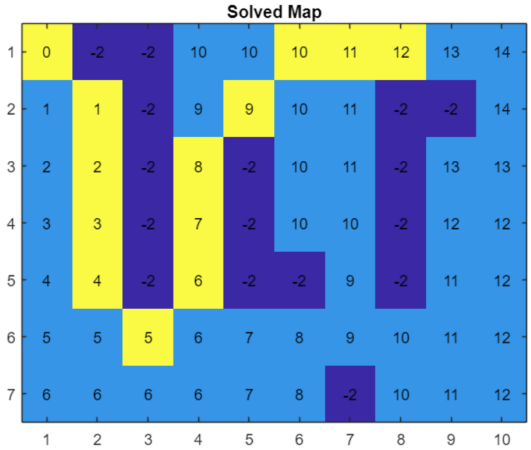

Figure 9: 8 adjacent square solution

It is immediate to observe how the 8-adjacent box algorithm returns a better result (fewer steps to the goal).

References

[1] Andrea Morghen. FSR2024. 2024. URL: https://github.com/Andremorgh/FSR2024.git.

[2] PRISMA UNINA. FSR_Slides_2024. 2024.