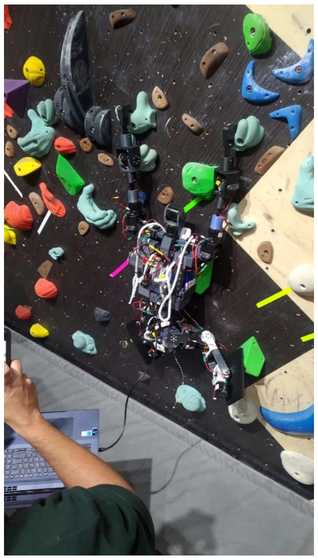
Intelligent Humanoid Robotics - Rock Climbing Robots

Jacky Baltes <jacky.baltes@ntnu.edu.tw>

Due Date: 23:59 on Monday, 15th December 2025

Important Updates

1. Page opened Wednesday, 31th October 2025
2. On 6th Nov. 2025, we conducted our first tests of our robot in a real climbing gym. No motion planning yet, but we managed to test several poses and holds.



Images from the first test with our new humanoid robot in a climbing gym

Content

The goal of this assignment is to implement a rock climbing robot.

Rock climbing is an extremely popular and complex sport (see Fig. 1). It requires strength, flexibility, balance, and motion planning abilities. As such, it is also a great benchmark problem for intelligent humanoid robotics.

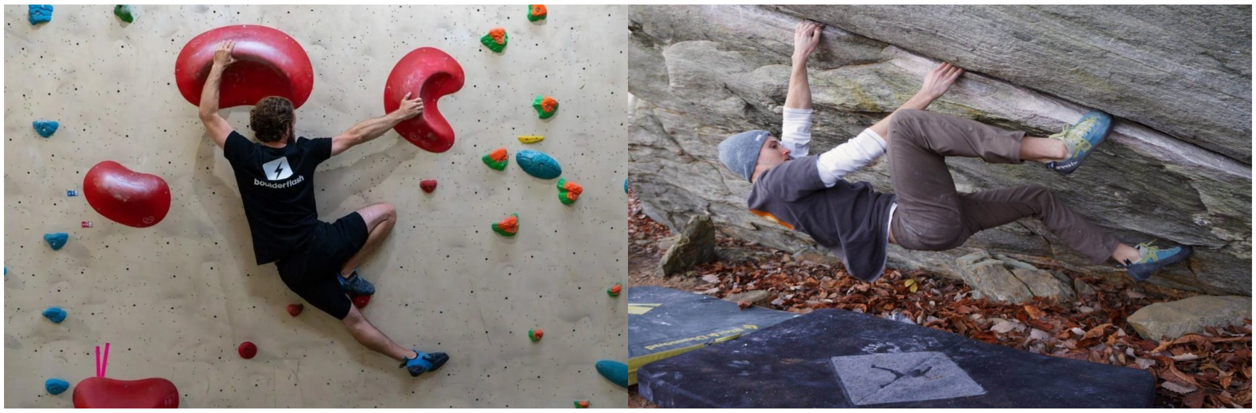


Fig 1: Humans can be extremely versatile and innovative motion planners when bouldering and rock climbing.

The robot will start with a global view of the climbing route. Based on this view of the climbing route, the robot create a motion path from the bottom to the top of the wall. The robot is then placed at the start position of the route. The robot will start to execute the motions until it reaches the top.

This assignment is an attempt at initial implementations of the necessary algorithms for a rock climbing robot. We break the assignment down into the following parts:

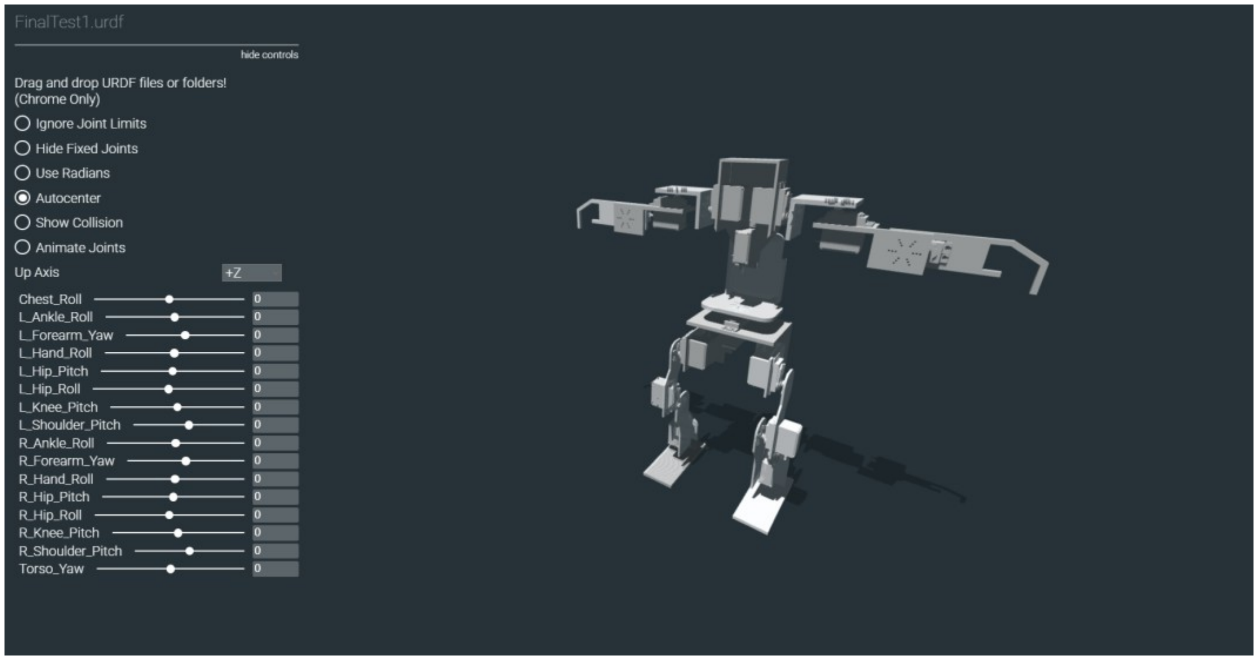
1. **Humanoid Robot Design**: Construct a simple, cheap, and efficient humanoid robot.
2. **Rock Climbing Robot Simulation**: Setup a simulation environment which simulates the floor, wall, various climbing holds, and various robots.
3. **Route Detection for Simulation and the Real World**: Implement a computer vision system that is able to extract hand and foot hold positions from an image. The figure below includes two real world views of a climbing route. Your system must be able to extract holds of a particular kind of color. Your system should output the type, position and orientation of your hold.
4. **Motion Planner**: Once you hae extracted and analyzed the route, you must plan a motion path for your robot to reach the top. The robot will start in a pre-defined starting position of the wall, where all four limbs are attached to a hold and the center of mass of the robot is withing the area spanned by the four holds.
5. **Climbing Control for Simulation and the Real World**: Once your robot has created a correct path, your robot must execute the path by moving the hands and feet to the correct location. After successfullly completing the climb in the MuJoCo simulation, you can try out your motions on the real robot.

These parts are described in more detail in the following subsections.

Physical Robot Construction (15 Marks Bonus)

The construction of this robot is based on standard RC servos and a Raspberry Pico 2 or similar embedded systems for control. Note that the robot will be placed at the start location, so it does not need to be able to walk to the wall. Furthermore, the robot can use external power, global vision, and external control which will simplify its design and greatly reduces its weight.

A simple design for such a robot, which can be constructed from plastic cut-offs or 3D printed is shown in the image below.


FinalTest1.urdf

hide controls

Drag and drop URDF files or folders!
(Chrome Only)

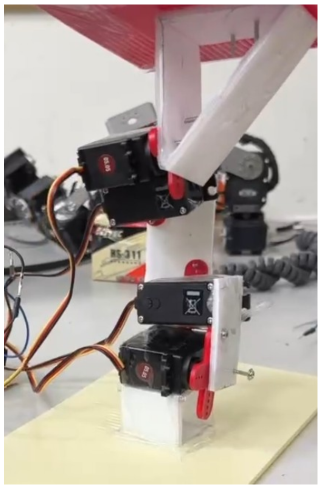
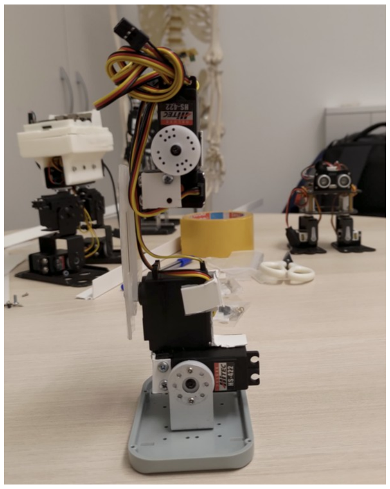
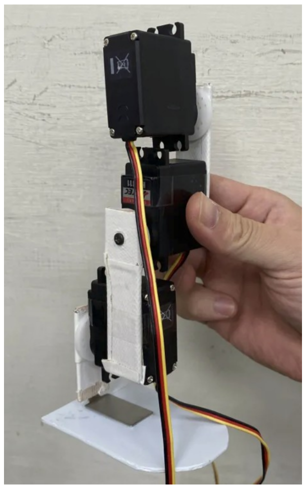
- Ignore Joint Limits
- Hide Fixed Joints
- Use Radians
- [✓ Autocenter]
- Show Collision
- Animate Joints

Up Axis +Z

Chest_Roll	0
L_Ankle_Roll	0
L_Forearm_Yaw	0
L_Hand_Roll	0
L_Hip_Pitch	0
L_Hip_Roll	0
L_Knee_Pitch	0
L_Shoulder_Pitch	0
R_Ankle_Roll	0
R_Forearm_Yaw	0
R_Hand_Roll	0
R_Hip_Pitch	0
R_Hip_Roll	0
R_Knee_Pitch	0
R_Shoulder_Pitch	0
Torso_Yaw	0

Sample design of small humanoid climbing robot. The design consists of 16 RC servo motors

Robots build by previous students are shown in the figure below.

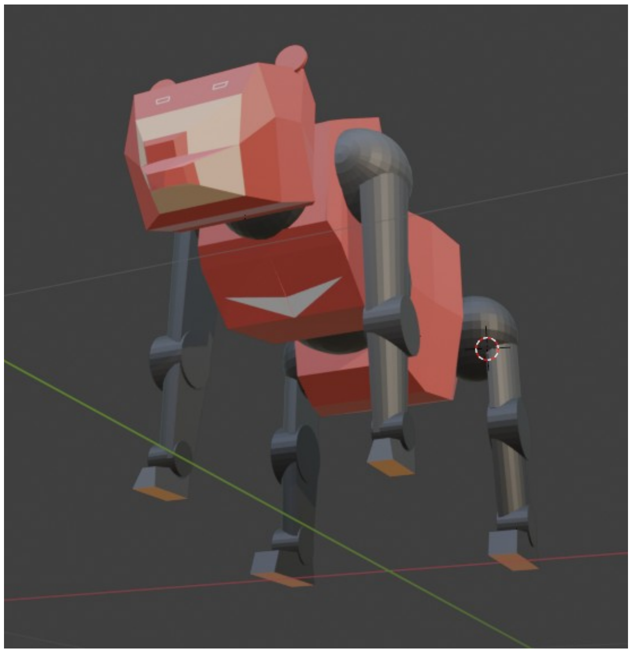
Pictures of robots build from cheap materials and standard RC servos built by students

Rock Climbing Robot Simulation (20 Marks Bonus)

This course uses the MuJoCo simulator. If you would like to use a different simulator (e.g., Gazebo or Isaac Lab), then you may do so, but you should discuss it with the course instructor first.

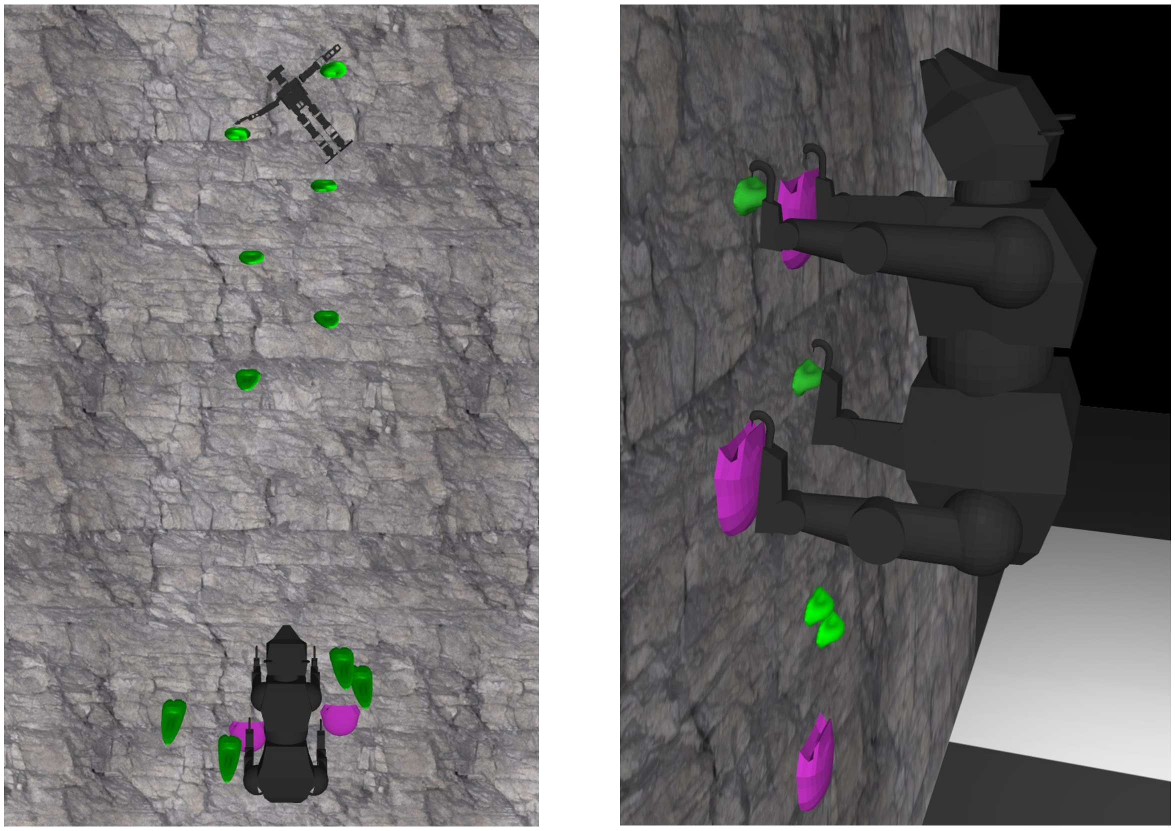
The rock-climbing-robot repository github repository includes ground floor, wall, and various holds to construct a rock climbing environment. Import and extend these assets and implement a program to simulate various climbing routes.

The lecture material already includes Taiwan Bear - a fantasy robot -, and OP3 – an accurate simulation of the Robotis OP3 robot. Import your own robot design into the simulator.



Taiwan robot bear and Robotis OP3 simulation

Create a standard pose for your robot with the robot extending the hands and feet to hang on to hand and foot holds.



OP3 and Taiwan Bear in the climbing wall (left) and close-up view of Taiwan Bear in the climbing wall

Collision Modeling: To speed up the simulation, MuJoCo does not check for arbitrary collisions between meshes. This means that most likely your robot will not collide with the hand holds. You can solve this problem in one of two ways:

1. Add simpler geometry (e.g., cubes, spheres, planes) to your model and the hand holds as collision boxes. You can see an example of this in the `op3.xml` file,

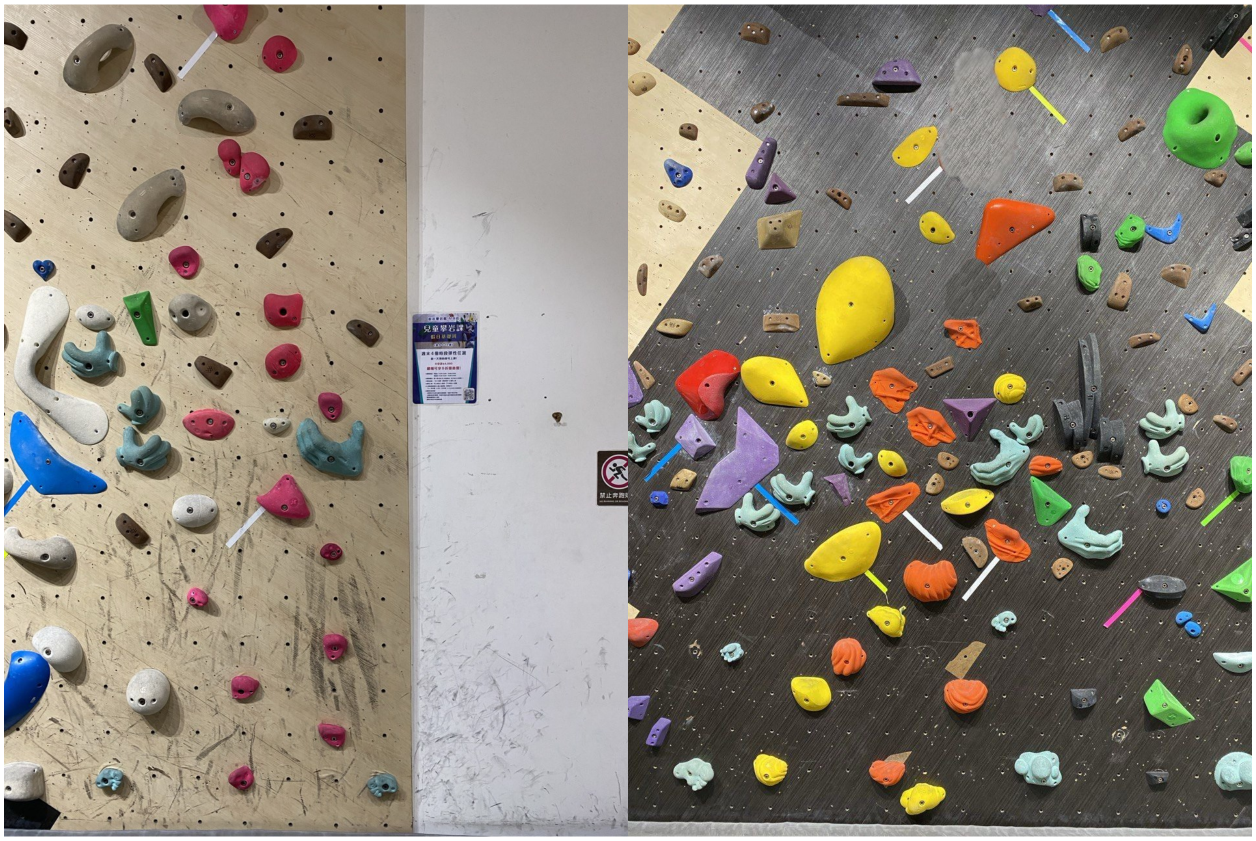
```
<joint name="r_hip_yaw" axis="0 0 -1"/>
    <geom mesh="rl1" class="visual"/>
    <geom mesh="rl1c" class="collision"/>
```

2. Use a tool such as CoACD to decompose your mesh into a collection of convex shapes.

Explain in your report which method you chose and explain the advantages and disadvantages of your approach.

Route Detection (30 Marks)

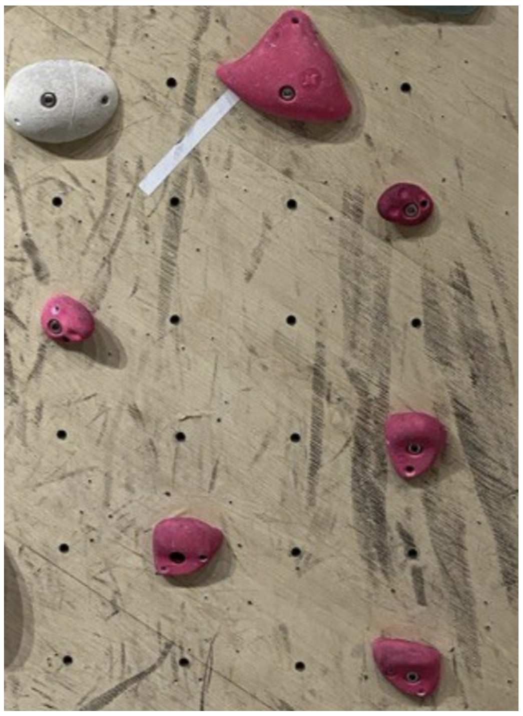
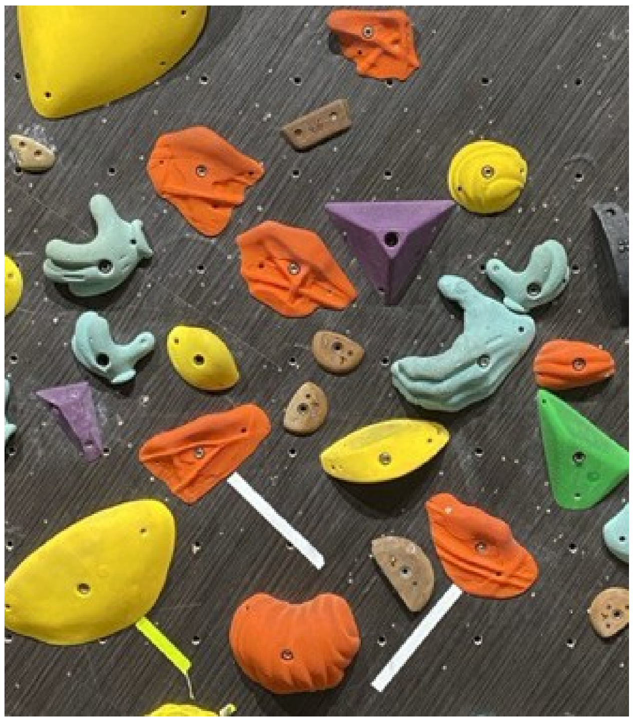
Your route detection will take as input an image of a climbing wall as shown in the figure below.



Sample images of climbing routes. The holds of a route are colour coded (e.g.:, left:magenta, right:orange).

Implement an algorithm that extracts the pixel location of the holds of a particular route for real world images. All input images will be taken with a view perpendicular to the wall.

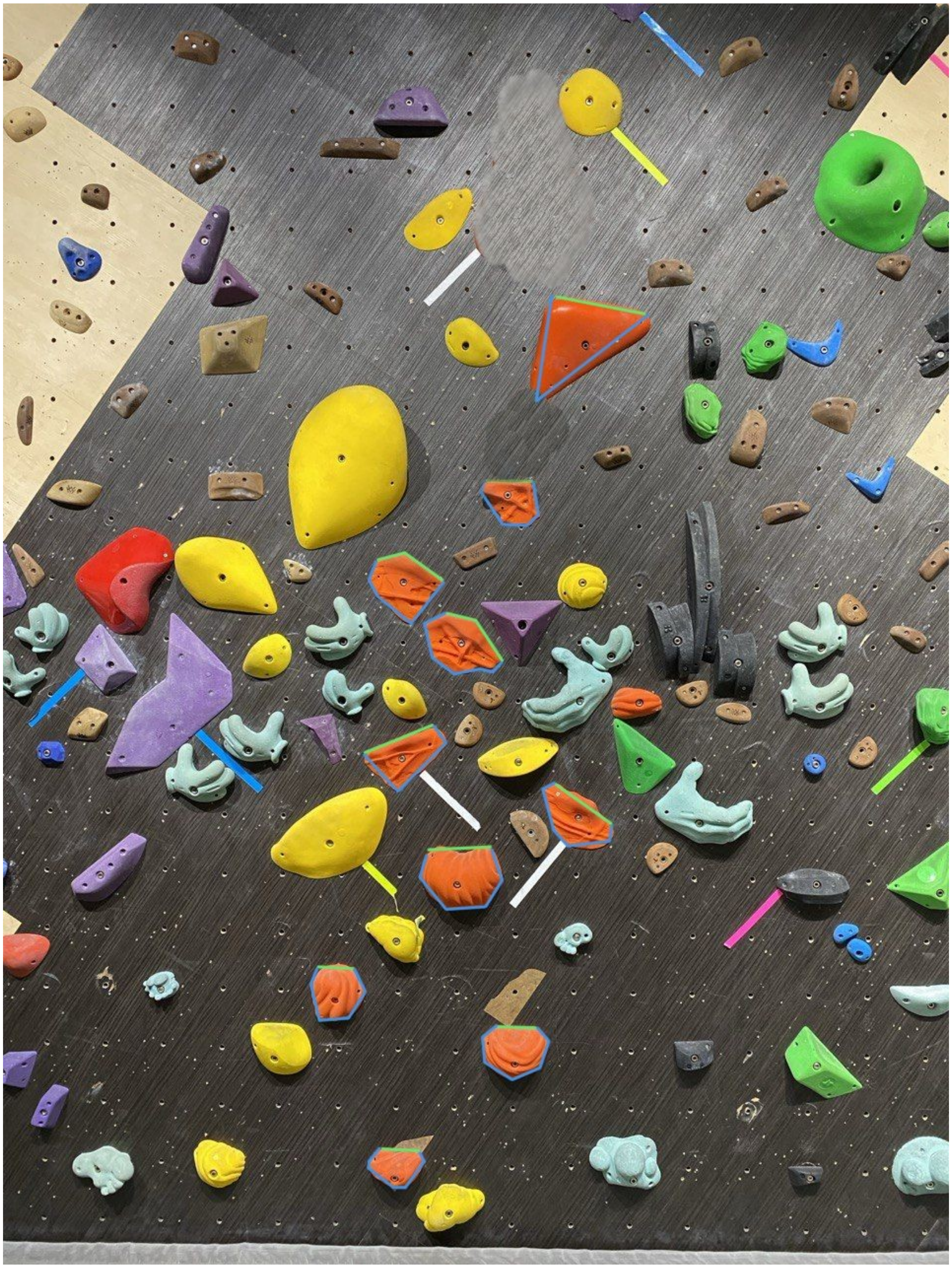
To initialize the system, your algorithm will be given a sample image of some part of the route which you can use to find suitable parameters for your algorithm.

Sample crops of the input image that your algorithm can use for calibration.

After your system has identified the holds of interest, it should extract the contour of the hold and return the position of the top of the hold as well as its slope/orientattion.

The output of your program for that image should be the pixel locations of the holds as well as the contour as shown in the image below.



Sample output of the hold detection and identification system.

Motion Planning (30 Marks)

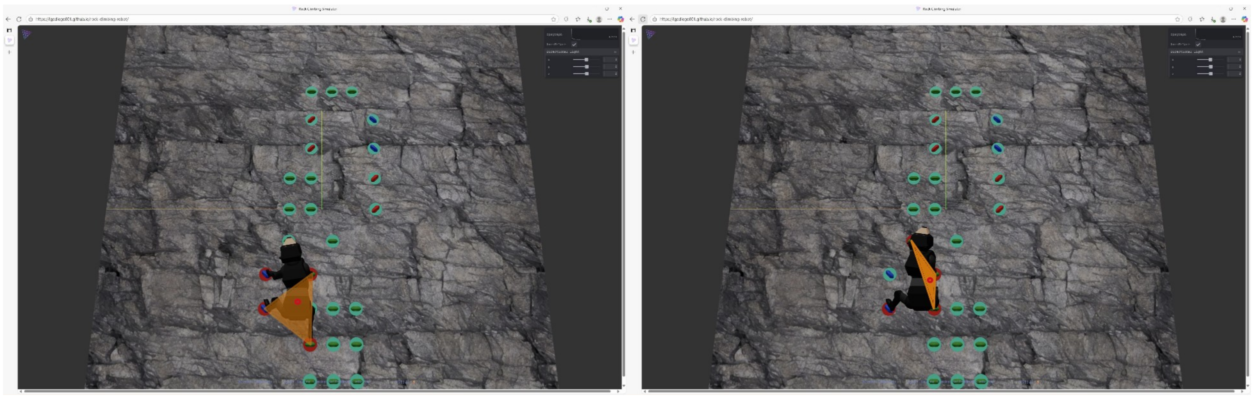
After successfully extract the route (i.e., type, position, and orientation of all holds) it is time to find a path from the starting/initial position to the top/goal position. So you must find a sequence of motions that when executed successfully will lead the robot through a series of holds to the top.

The planning problem is one of the fundamental problems in artificial intelligence with a long history of research. The crucial problem is the fact that the search explodes exponentially in the number of steps/actions necessary to reach the goal.

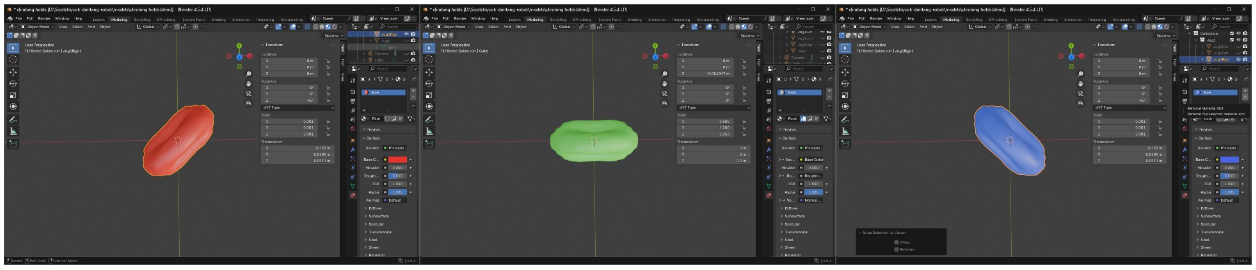
The A* algorithm is one of the most well known algorithms for solving planning problems, because domain knowledge to speed up the planner can be incorporated using an admissible heuristic function. More recently randomised algorithms, e.g., rapidly exploring random trees (RRTs), haven proven to be effective.

There is an almost infintite number of motions that a human climber can perform to reach the top, such as dynamic pogo moves, bicycle holds with the legs, dyno moves jumoing to the next hold, ...). This makes the search for possible action sequences astronomically large.

Simple climbing strategy: To make this problem manageable and to reduce the size of the search space, we constrain our climbing robot to a simple climbing strategy. At each step, the robot has at least three limbs attached to the wall. The three holds form a triangle and the center of mass of the robot is always within this triangle. The free limb moves to a new hold (i.e., the hand to a new hand hold or a foot to a new foot hold). If the hold is suitable to be held by the robot, then the free limb will attach to the hold. The robot can then detach some other limb and move its center of mass to a new location. Then the new free limb will attach to a new hold and the process iterates.


Taiwan bear executing the simple climbing strategy in the rock climbing simulator

Furthermore, we limit the holds to only three types: left, center, and right holds. The center hold allows a robot to grab the hold with its hands if its center of mass is below the hold. The robot can grab the left and right holds only if the center of mass is below the hold and to the right and left of the hold respectively.


The three types of climbing holds (left/red is a hold for left, center/green is in any direction, and right/blue is a hold to the right)

All three types can be used as foot holds as long as the center of mass of the robot is above the hold.

An advantage of this strategy is that it is only based on geometric properties of the robot and the holds. So you do not need to worry about the dynamic properties (e.g., friction of the holds maximum torque of the motors, etc.). To implement the path planner it is sufficient to compute forward and inverse kinematics for the limbs of the climbing robot.

Implement an A* or RRT based motion planner that searches through the action space using the limited climbing strategy described above.

Plan Execution in Simulation (20 Marks)

After your planner has created a new plan to reach the top, the plan executor will execute the motions and supervise successful execution.

Errors due to noise in the vision system may lead to the robot missing a hold or hitting an obstacle. To avoid this problem, your executor must check the contact forces on the hands and feet to make sure that the hold has been reached before moving on to the next motion.

Plan Execution in the Real World (15 Marks Bonus)

For the real robot, the simple RC servos do not provide position or torque feedback, so all the plan executor can do is execute the motions. Implement an emergency brake in your system, that will stop the robot, if a human operator detects an error.

Submission

All your files and analysis should go into a directory labeled `<user_id>_rock_climbing`, where `<user_id>` is your own user id.

Add a README.txt file into the directory, which shows: (a) what other sources you used apart from the lecture material used in class during your work on the assignment, and (b) how to compile and run your program, and (c) any interesting features and extensions of your assignment.

Add a file `report.pdf`, which describes your project and shows the results of your evaluations. Discuss the results. Were the results expected or did they surprise you? Did the results highlight shortcomings in the system? Do you have any ideas for how to fix remaining problems?

Submit your assignment via email to jacky.baltes@ntnu.edu.tw. You can submit a compressed directory of your source code and analysis directly if the resulting file is less than 2 Megabyte. If the file is larger than create a google drive folder and send me a sharable link to the folder to my email.