RoVi Final Project

Version 1.0

Fall 2022

1 Background

Pick and Place tasks are one of the most frequent tasks executed by robots. In typical production setups the pick and place locations are well known in advance and do not change, meaning, the parts are placed in specialised holders, preventing them to change pose although an disturbance force may occur. In the majority of the cases the part holders add additional complexity and increase the cost of the application. In recent years there is a strive to make production setups flexible to allow for quick changeover between products. Therefore there is a need for vision based localisation solutions in order to overcome the limitations of fixed holders and to make applications more flexible.

2 Overview

In this project you are asked to design such a pick and place solution (in a simulated environment). The task can be decomposed into three major component, the robotics parts (see Section 3), the vision part (see Sections 4) and the combination of both (see Section 5).

3 Pick and Place

Before combining the robotic part with vision and executing the grasping operation, you are asked in this section, to prepare the workcell and check your trajectory planning algorithms.

3.1 Workcell setup

In order to start the simulation, you first have to setup the work environment. All of the needed models for this step are provided in the zip file, which can be downloaded from itslearning.

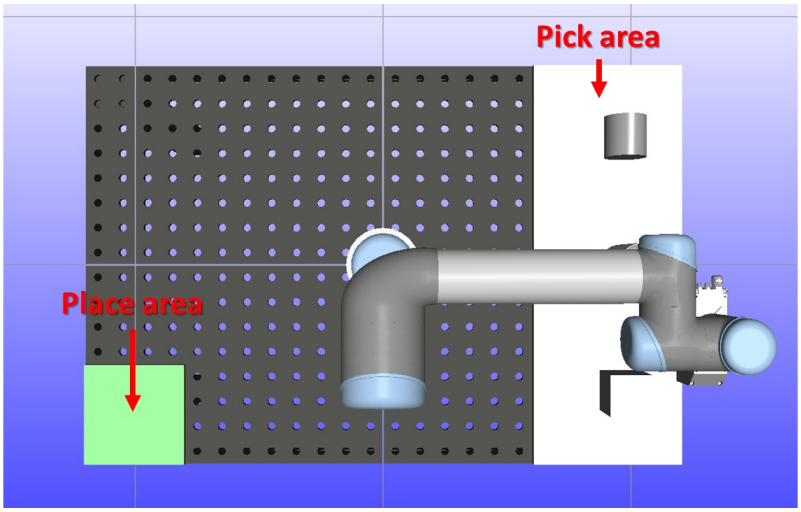


Figure 1: Pick and place locations.

In the zip file you will find a folder called `scene`, containing the cell definition and some additional folders with the robot, gripper, table and object representations. The requirements of this section are:

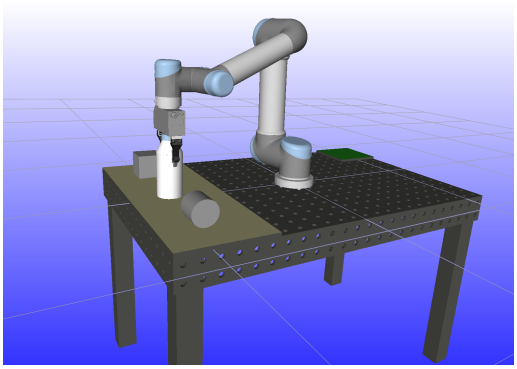
- Add the base mount of the robot to the scene,
- add the robot and attach the gripper to the Tcp frame of the robot, so it will be included in the kinematic chain,
- define the robot Home pos,
- add the test objects to the predefined pick area (Fig. 1) in the scene.

After fulfilling all of the requirements, the scene is ready for the next step.

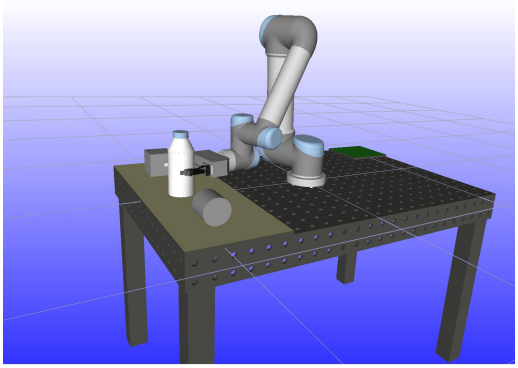
3.2 Testing the reachability of the manipulator

Similar as in *programming exercise 5*, you are asked to determine the best mounting position (base position), so that the robot can reach both the pick and place locations in the scene with as many collision free kinematic configurations as possible. Requirements:

- Consider two grasping possibilities, from the top (see Fig. 2 a) and from the side (see Fig. 2 b).
- Perform the study based on 30 - 60 different mounting positions for each grasping possibility, you can also vary the position of the object - it does not have to be static.



(a) Grasp from the top



(b) Grasp from the side

Figure 2: Grasp configurations

- The results should be presented in a graphical form (preferably with reachability areas), and statistical evaluation (graph representing the relation between moving the base and number of collision free representations.)

After you conclude the study, pick the best location and give a description why this is the best position.

3.3 Robot motion planning

In order to execute the pick and place action, you are asked to calculate the robot trajectories for this specific task. To do so use / implement the following algorithms:

- Point to Point interpolator, you should select at least 6 points to interpolate between in order to fulfill the pick and place task.
- Point to Point interpolator with parabolic blend.
- Use the RRT connect algorithm to plan the trajectory. In order for the RRT connect to work, you have to introduce an obstacle into the workcell. In order to do so load the second workcell provided by us (workcell_with_obstacles). Do not forget to place the robot at the location derived by the reachability analysis.

Implement all three approaches and test their performance with at least 3 different pick locations of the object in the scene. The place location can be the same for all approaches.

3.4 Evaluation and presentation of the results

- Give a written description of the implementation steps and selected parameters, for each algorithm.
- Statistically evaluate the performance of each algorithm (execution time vs. accuracy - perform at least 50 executions per algorithm and pick location).
- Present your results and findings in a graphical way (trajectory plots, performance charts).
- Provide a measure and a description of your decision, which algorithm is best for the presented task.

4 Vision

The task on the vision side is to develop two pose estimation methods. You need to choose two of the following approaches: Sparse stereo, Dense stereo, Simulated depth sensor and image based (you can not choose both dense stereo and simulated depth sensor). You then need to, *for each of the two methods independently*, implement the method, choose objects suitable for the method, place the sensor at a realistic location, potentially optimise the parameters of the method and then evaluate the robustness of the method (sensitivity analysis).

4.1 Methods

You have to implement two of the following four pose estimation approaches (you can not choose both M1 and M2). All of them will have some sensor input (e.g., single image, pair of stereo images, point cloud) in addition to some object model/template/... and will as a result produce an object pose relative to a sensible coordinate system (e.g., the world coordinate system of your workcell).

M1 Dense stereo: A method that gets images from the left and the right camera, computes disparity information and then computes the object pose based on that.

The pipeline here would typically be (you decide on which steps to do or not to do and which concrete choices to make here):

- Get images
- Rectify and undistort images (if necessary)

- Compute disparity map
- Compute point cloud
- Filter/Segment point cloud
- Use global pose estimation to find the object pose
- Use local pose estimation to refine the object pose

M2 Simulated depth sensor: A method that gets a point cloud from the simulated depth sensor and then computes the object pose based on that.

The pipeline here would typically be (you decide on which steps to do or not to do and which concrete choices to make here):

- Get point cloud
- Filter/Segment point cloud
- Use global pose estimation to find the object pose
- Use local pose estimation to refine the object pose

M3 Sparse stereo: A method that gets images from the left and the right camera, computes a relevant feature in the two images, triangulates these features and then computes the object pose based on that.

The pipeline here would typically be (you decide on which steps to do or not to do and which concrete choices to make here):

- Get images
- Rectify and undistort images (if necessary)
- Find the relevant features in the two images (e.g., find the centre of the red ball, find the corners of the square, ...)
- Match up the features from the left and the right image
- Triangulate the features
- Use the 3D feature locations to find the object pose (e.g., 3D centre of the ball, position and orientation of the square plane, ...)

The initial idea here was to use application specific features (e.g., 3D centre of the ball, position and orientation of the square plane, ...), in case you want to use generic features (e.g., SIFT) please note that you need to relate your features to the object reference frame so that you can then later use that relationship to compute the new object pose from the feature locations.

M4 Image based: A method that gets an image from the camera and then computes the object pose based on that.

The pipeline here would typically be (you decide on which steps to do or not to do and which concrete choices to make here):

- Get image
- Undistort images (if necessary)
- Use the image based pose estimation approach to find the object pose

4.2 Setup

Test the two implemented methods with likely object positions in the scene. You should make sure to choose objects here that are suitable to your method, that you place the camera in a suitable location relative to the area in which your object will appear (the camera should be statically placed for each method, it should not move depending on specific object position) and that you setup the parameters of your methods (e.g., stereo parameters (e.g., disparity range), pose estimation parameters (e.g., RANSAC iteration count)) according to the need in your scene.

4.3 Evaluation

Once you have implemented and setup your methods in a reasonable way it is time to evaluate the performance of the method. The idea here is to evaluate the methods performance under different conditions and to see how much you can stress the method before it breaks.

Performance here can typically be measured in two ways:

- Do you get a reasonable pose estimate at all or is the result not even close to the real object position. You want to measure this as a binary outcome and then summarise it as a percentage of cases the method works. (You could now make a manual decision for each pose estimate here if you feel the outcome if reasonable or not. Another more practical approach might be to define bounds on the pose error and to decide based on these.)

- The pose error when the pose estimate works (is close to the real pose). Here you want to filter out the cases where you above decided that the pose is not good and then you want to calculate the distance to the real pose. Typically you have to do this independently for position and orientation.

Many of the processes here are deterministic if the input is deterministic (which it is in our case because of the nature of the simulator/rendering). You therefore need to add some indeterminism to the process. Typically this is done in the form of noise. Noise here can be added in different forms:

- Image noise. Adding Gaussian noise to the images is a possible way. This is applicable to M1, M3 and M4.
- Additive Gaussian noise added to all the computed points. This is possible for M1 (the point cloud produced by stereo), M2 (the point cloud coming from the sensor), M3 (the sparse 3D points produced by sparse stereo).

Choose your way in which you want to add noise to your process. The magnitude of the noise is important here. Try out different amounts of noise until you break the algorithm. Then show a graph that shows the performance of the algorithm degrading when the amount of noise is increased to the breaking point.

To show that this is robust to different object positions you also want to do this for multiple different object positions and either show this for the different ones individually or also integrate multiple positions into averaged results.

4.4 Presentation of results / documentation

The report needs to contain the following parts for both implemented methods

- A description of the implemented method. Ideally with comments on design choices (why where certain things done or not done?)
- A description on choices made for the setup. Ideally with comments on the choices (why where certain things done or not done?)
- A detailed description on the evaluation performed (the setup of the experiments). Ideally with comments on the choices (why where certain things done or not done?)
- A detailed description on the outcome of the experiments, displaying the results in a suitable way and a discussion on the results (what do the results mean? what can we take away from it? ...).
- Compare the results of the two methods. Which one would you choose for which reasons?

5 Combining pose estimation with pick and place execution

5.1 Integration

Once you have implemented the solution for the robotics and the vision part above it is time to integrate the two parts.

The idea here is that you replace the hardcoded pick locations that you have been using in the robotics part with poses coming from *one* of the pose estimation methods you implemented in the vision part.

5.2 Documentation

Please describe how you integrated the two parts. Where there any challenges? Any important parts you want to point out?

Document the performance of the combined system by trying to pick and place the object from multiple different locations. (A low count of locations is OK here since you hopefully did an in depth evaluation for the robotics and vision part individually.)

6 Formalities

Deadline for the project is December, 23^{rd} , 9am. The project report + zip file should be delivered via Digital Exam in groups of two persons, guide under "Submitting written assignments in Digital Exam" (here). Mails will not be accepted as hand-in. The formal requirements for the hand are the following:

- The report must contain a section explaining in detail which team-member contributed what to the project. (E.g., Who implemented which methods? Wo designed and run which experiments? Who wrote which part of the text?). If this part is not provided the overall report will be graded as zero points independent of any other factors.
- The report must contain the parts mentioned in Sections 3.4, 4.4 and 5.2 in addition to some general report parts and glue binding things together.
- Report clearly explaining the algorithms used; the motivation behind the choices and the tests performed. The report is to be between 2000 and 6000 words.
- A zip file containing the source code developed in the project and a plain text-file explaining the content of the files, how to compile and how to run the tests.

The project is to be solved in groups of two persons. Groups are NOT allowed to copy text or code from each other! It is completely OK to discuss with each other and inspire each other. In case copying (of text or code) is found both groups will fail the course and be reported to the university officials! If code snippets from the exercise solutions on itslearning (or other public locations) are used, make sure they are properly referenced in your report.