Identify Obstacles And Grab Items By Robot Arms

A project of the 2017 Robotics Course of the School of Information Science and Technology (SIST) of ShanghaiTech University

Haihao Zhu
zhuhh2@shanghaitech.edu.cn

Hudie Gu
guhd@shanghaitech.edu.cn

Yi Yang
yangyi1@shanghaitech.edu.cn

Abstract

This project is based on Dobot Magician Robot Arm and Ros, we plan to create a model by usig URDF file in Ros invironment, and make the model move along the real robot.

1. Introduction

1.1. Dobot

The Dobot Magician Robot Arm is multi-task robot arm that has black and white body with consistent qualityat fingertipsUnibody design, unbox and enjoy right away Designed for desktop, safe and easily lifted. It contains a Vacuum pump Kit , Gripper, Wirting and Drawing Kit, 3D Printing Kit, and other accessories.It has many features like writing, drawing, 3D printing and even playing chessing with human.

Dobot Magician is an All-in-One Robot arm for Education. It can be controlled by PC, phone, gesture or voice since it can communicate by USB, Wifi and Bluetooch.Dobot Magician has 4 axes while its maximum reach is 320mm and its position repeatability(control) is 0.2mm. Dobot magician has 13 extension ports, 1 programmable key and 2MB offline command storage. One of dobots main characters is its various endeffectors, such as 3D Printer Kit, pen holder, vacuum suction cap, gripper and laser. The softwares for dobot magician are DobotStudio(for Windows XP, win7 SPI, win8/win10, mac osx 10.10 and mac osx 10.11, mac osx 10.12), Repetier Host, GrblController3.6, DobotBlockly (Visual Programing editor) while the software develop kit is Communication Protocol, Dobot Program Library. In ROS, dobot doesnt have much function except controlling the arm by keyboard using package.

Dobot magician has 13 extension ports, 1 programmable key and 2MB offline command storage. One of dobots main characters is its various endeffectors, such as 3D Printer Kit, pen holder, vacuum suction cap, gripper and laser.The softwares for dobot magician are DobotStudio(for Windows XP, win7 SPI, win8/win10, mac osx 10.10 and mac osx 10.11, mac osx 10.12), Repetier Host, GrblController3.6, DobotBlockly (Visual Programing editor) while the software develop kit is Communication Protocol, Dobot Program Library. In ROS, dobot doesnt have much function except controlling the arm by keyboard using package.

Figure 1. Dobot Magician Arm

The figure below is what the Dobot arm looks like.

1.2. URDF

URDF means Unified Robot Description Format, it is Kinematic and basic physics description of a robot in XML

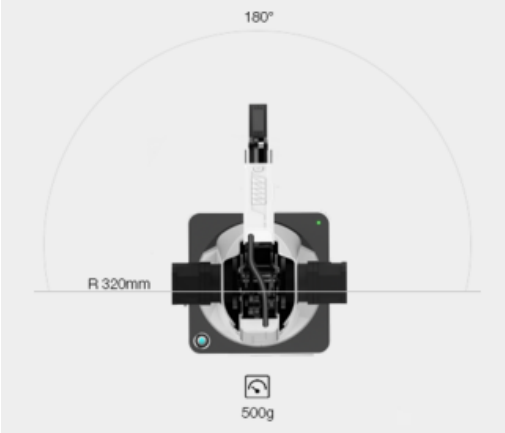

Figure 2. Dobot Magician Arm

Figure 3. Dobot Magician Arm

file format. It mainly consists of tags like links , joint and material. Many of the coolest and most useful capabilities of ROS and its community involve things like collision checking and dynamic path planning. Its frequently useful to have a code-independent, human-readable way to describe the geometry of robots and their cells. Think of it like a textual CAD description: part-one is 1 meter left of part-two and has the following triangle-mesh for display purposes. The Unified Robot Description Format (URDF) is the most popular of these formats today. This module will walk you through creating a simple robot cell that well expand upon and use for practical purposes later.

2. State of the art

Since the initial goal of this project is trying to identify obstacles and grasp items by robot arms, but situation changes when the initial robot arm is broken. Then we changed our goal to model the new Dobot robot arm and make it move along the real Maigician Dobot Arm, which has no State-Of-The-Art part, so we list the content in the intermidiate report.

2.1. Haihao Zhu

1. **Motion Planning of Humanoid Robot Arm for grasping task** [3]

   This paper demonstrates a new method for computing numerical solution to the motion planning of humanoid robot arm. The method is based on combination of two nonlinear programming techniques which are Forward recursion formula and FBS method.

2. **Autonomous vision-guided bi-manual grasping and manipulation** [4]

   This paper describes the implementation, demonstration and evaluation of a variety of autonomous, vision-guided manipulation capabilities, using a dual-arm Baxter robot. It starts from the case that human operator moves the master arm and slave arm follows the master arm, and then, it combines an image-based visual servoing scheme with the first case to perform dual-arm manipulation without human intervention.

3. **Arm grasping for mobile robot transportation using Kinect sensor and kinematic analysis** [1]

   In this paper, we describe how the grasping and placing strategies for 6 DOF arms of a H20 mobile robot can be supported to achieve a high precision performance for a safe transportation. An accurate kinematic model has bean used in this paper to find safe path from one pose to another precisely.

4. **A new method for mobile robot arm blind grasping using ultrasonic sensors and Artificial Neural Networks** [2]

   The paper presents a new method to realize mobile robot arm grasping in indoor laboratory environments. This method adopts a blind strategy, which does not need the robot arms be mounted any kind sensors and avoid calculating the complex kinematic equations of the arms.

   The method includes:

   (a) two robot on-board ultrasonic sensors in base are utilized to measure the distances between the robot base and the front arm grasping tables;

(b) an Artificial Neural Networks (ANN) is proposed to learn/establish the nonlinear relationship between the ultrasonic distances and the joint controlling values. After executing the training step using sampling data, the ANN can forecast/generate the next-step joint controlling values fast and accurately by inputting a new pair of real-time ultrasonic measured distances;

(c) to let the blind strategy matching with the transportation process, an arm controlling component with user interfaces is developed;

(d) a method named training arm is adopted to prepare the training data for the training procedure of the ANN model.

Finally, an experiment proves that the proposed strategy has good performance in both of the accuracy and the real-time computation, which can be applied to the real-time arm operations for the mobile robot transportation in laboratory automation.

5. **find-object**

Find-object is a Simple Qt interface to try OpenCV implementations of SIFT, SURF, FAST, BRIEF and other feature detectors and descriptors.

it have many features like:

(a) You can change any parameters at runtime, make it easier to test feature detectors and descriptors without always recompiling.

(b) Detectors/descriptors supported (from OpenCV): BRIEF, Dense, FAST, GoodFeaturesToTrack, MSER, ORB, SIFT, STAR, SURF, FREAK and BRISK.

(c) Sample code with the OpenCV C++ interface below...

(d) For an example of an application using SURF descriptors: see my project RTAB-Map (an appearance-based loop closure detector for SLAM).

2.2. Yi Yang

1. Dynamic Sensor-Based Control of Robots with Visual Feedback

This paper describe the formulation of sensory feedback models for systems which incorporate complex mappings between robot, sensor, and world coordinate frames. These models explicitly address the use of sensory features to define hierarchical control structures, and the definition of control strategies which achieve consistent dynamic performance. Specific simulation studies examine how adaptive control may be used to control a robot based on image feature reference and feedback signals.

2. Vision-based Motion Planning For A Robot Arm Using Topology Representing Networks

This paper describe the concept of the Perceptual Control Manifold and the Topology Representing Network. By exploiting the topology preserving features of the neural network, path planning strategies defined on the TRN lead to flexible obstacle avoidance. The practical feasibility of the approach is demonstrated by the results of simulation with a PUMA robot and experiments with a Mitsubishi Robot.

3. A Framework for Robot Motion Planning with Sensor Constraints

This paper propose a motion planning framework that achieves this with the help of a space called the perceptual control manifold (PCM) defined on the product of the robot configuration space and an image-based feature space. They show how the task of intercepting a moving target can be mapped to the PCM, using image feature trajectories of the robot end-effector and the moving target. This leads to the generation of motion plans that satisfy various constraints and optimality criteria derived from the robot kinematics, the control system, and the sensing mechanism, specific interception tasks are analyzed to illustrate this vision-based planning technique.

4. Motion Planning of a Pneumatic Robot Using a Neural Network

This paper present a framework for sensor-based robot motion planning that uses learning to handle arbitrarily configured sensors and robots. Autonomous robotics requires the generation of motion plans for achieving goals while satisfying environmental constraints. Classical motion planning is defined on a configuration space which is generally assumed to be known, implying the complete knowledge of both the robot kinematics as well as knowledge of the obstacles in the configuration space. Uncertainty, however, is prevalent, which makes such motion planning techniques inadequate for practical purposes. Sensors such as cameras can help in overcoming uncertainties but require proper utilization of sensor feedback for this purpose. A robot motion plan should incorporate constraints from the sensor system as well as criteria for optimizing the sensor feedback. However, in most motion planning approaches, sensing is decoupled from planning. A framework for motion planning was proposed that considers sensors as an integral part of the: definition of the motion goal. The approach is based on

the concept of a Perceptual Control Manifold (PCM), defined on the product of the robot configuration space and sensor space. The PCM provides a flexible way of developing motion plans that exploit sensors effectively. However, there are robotic systems, where the PCM cannot be derived analytically, since the exact mathematical relationship between configuration space, sensor space, and control signals is not known. Even if the PCM is known analytically, motion planning may require the tedious and error prone process of calibration of both the kinematic and imaging parameters of the system. Instead of using the analytical expressions for deriving the PCM, they therefore propose the use of a self-organizing neural network to learn the topology of this manifold. They first develop the general PCM concept, then describe the Topology Representing Network (TRN) algorithm they use to approximate the PCM and a diffusion-based path planning strategy which can be employed in conjunction with the TRN. Path control and flexible obstacle avoidance demonstrate the feasibility of this approach for motion planning in a realistic environment and illustrate the potential for further robotic applications.

5. *usb_cam*

   The usb_cam package, usually together used with cv_camera package, is a driver used to make the camera capture the real environment and use the data transmitted by the camera to form a picture. It will create a usb_cam_node. The usb_cam_node interfaces with standard USB cameras using libusb_cam and publishes images as sensor_msgs::Image. Uses the image_transport library to allow compressed image transport. The published topics is ¡camera_name¿/image. It has lots of parameters, video_device (string, default: "/dev/video0") to choose the device the camera is on; image_width (integer, default: 640) and image_height (integer, default: 480) to set image width and image height; pixel_format (string, default: "mjpeg") to change the picture format which possible values are mjpeg, yuyv, uyvy; io_method (string, default: "mmap") to change the input and output method which possible values are mmap, read, userptr; camera_frame_id (string, default: "head_camera") to set the camera's tf frame; framerate (integer, default: 30) to change the required framerate; contrast (integer, default: 32) to set the contrast of video image (0-255); brightness (integer, default: 32) to set brightness of video image (0-255); saturation (integer, default: 32) to set saturation of video image (0-255); sharpness (integer, default: 22) to set sharpness of video image (0-255); autofocus (boolean, default: false) to determine whether enable camera's autofocus or not; focus (integer, default: 51) to set the focus of the camera (0=at infinity) if autofocus is disabled. camera_info_url (string, default:) is an url to the camera calibration file that will be read by the CameraInfoManager class; camera_name (string, default: head_camera) is the camera name and must match the name in the camera calibration. The related packages cv_camera is used to supports image capture from usb cameras using OpenCV. To use the package, first git clone https://github.com/bosch-ros-pkg/usb_cam.git usb_cam to clone the usb_cam package. Then build the package. Afterwards, build .launch file. An example is like behind:

```
$<launch>
$  <node name="usb_cam" pkg="usb_cam"
        type="usb_cam_node" output="screen" >
$    <param name="video_device" value="/dev/
$    <param name="image_width" value="640" />
$    <param name="image_height" value="480" >
$    <param name="pixel_format" value="mjpeg
$    <param name="camera_frame_id" value="us
$    <param name="io_method" value="mmap"/>
$  </node>
$  <node name="image_view" pkg="image_view"
        respawn="false" output="screen">
$    <remap from="image" to="/usb_cam/image_
$    <param name="autosize" value="true" />
$  </node>
$</launch>
```

Finally, use roslaunch usb_cam.launch to launch.

2.3. Hudie Gu

1. Vision based adaptive grasping of a humanoid robot arm

   This paper presents a motion planning and control design of a humanoid robot arm for vision-based grasping in an obstructed environment. This study proposes a design for safe operation of the robot arm in an unknown environment, and practical experiments show that the six degree- of-freedom robot arm can effectively avoid obstacles and complete the grasping task.

2. The intelligent robot arm based on sense of sight

   This paper designed This paper designs an intelligent robot arm system based on visual sensor, which can achieve the object's searching and positioning through vision system. In the scheme design, a camera is install above the work area of robot arm, which captures the image of work area in real time. Image segmentation and feature extraction is done by MATLAB to recognize objects.

3. Optimal trajectory generation for energy consumption minimization and moving obstacle avoidance of a 4DOF robot arm

   In this paper, trajectory generation for a 4 DOF arm of SURENA III humanoid robot with the purpose of optimizing energy and avoiding a moving obstacle is presented. This paper use Lagrange approach to model dynamic behavior of a robotic manipulator and a Genetic Algorithm to optimize robot's movements.

4. Multi objective optimization of humanoid robot arm motion for obstacle avoidance

   This paper proposed the neural controllers for mobile humanoid robot arm in presence of obstacle and optimizing time, distance and acceleration simultaneously.

   The optimization problem is the arm motion generation for obstacle avoidance. To solve it, this research utilized a feedforward neural network(FFNN)while using Laser Range Finder to determine the position of the target. Both hands generate a set of optimized FFNN while each FFNN receives three input.

   To solve obstacle avoidance, this research divided the obstacle area in lateral plan in 6 parts and a single pre-evolved neural controller is generated for each section. Then the robot determines the partitions that the obstacle covers based on the obstacle position and size. The robot arm motion is generated based on the specific neural controller reaching the goal position while avoiding the obstacle.

5. Moveit!

   MoveIt! is state of the art software for mobile manipulation, incorporating the latest advances in motion planning, manipulation, 3D perception, kinematics, control and navigation.

   Primary users can use this interface both through C++ and Python through the move_ group_ interface.

   MoveIt! comes with a plugin for the ROS Visualizer (RViz). The plugin allows users to setup scenes in which the robot will work, generate plans, visualize the output and interact directly with a visualized robot. In this plugin, there are four different visualizations active:

   1.The start state for motion planning;

   2.The goal state for motion planning;

   3.The robots configuration in the planning scene/ planning environment;

   4.The planned path for the robot.

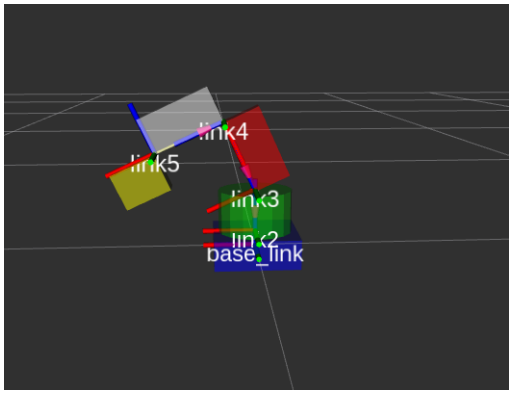

Figure 4. URDF model of the Dobot Magician Arm

3. System Description

In this project, We intend to realize the following three goals. First of all, we plan to control the robot-arm by human with keyboard manually. Afterwards, we want the robot-arm to automatically seek and fetch the object in the same plane. Besides, we hope the robot-arm can achieve the function that moving the object to a specific location while recognizing and avoiding obstacles.

We will use Ros, moveit, and maybe find-object to achieve the goal above. We will use Ros to control the whole robot arm, and use Moveit package to do motion planning, manipulation, and even collision detection. If we have time, we will try to use find-object package to find multiple object.

For motion planning, we will probably use OMPL(Open Motion Planning Library) algorithm which is the default of Moveit!. For Collision Checking, we will use FCL(Flexible Collision Library) of Moveit! to find check whether there exists collision in planning pat based on the OctoMap generated by 3D Perception plugin.

For computer vision task, we will use apriltags to find where the tube are relavant to camera.

4. System Evaluation

4.1. URDF model

We have built a URDF model by creating URDF file with real size, To simplify, we use cylinder and box to represent links of the arm. It looks like below:

We can see from the figure 1, dobot magician robot arm has a fice links including base link and 4 joint between each of them. To achieve our goal, we need to get the joint message of each joint of the real robot and set it into the joint of URDF model by using $joint_state_publisher$ package in URDF file.

4.2. Dobot API

The main point is how to get joint message of the real robot. We can get it by Dobot API. Since it is ugly to put the code here, I have put the code we write in my git repository [].

5. Conclusion

Finaly, we have achieved our goal that the mode move along the real robot when we press keyboard to control the real robot. The demo can be found in our webpage of this project []

References

[1] M. M. Ali, H. Liu, R. Stoll, and K. Thurow. Arm grasping for mobile robot transportation using kinect sensor and kinematic analysis. In *2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings*, pages 516–521, May 2015.

[2] H. Liu, N. Stoll, S. Junginger, and K. Thurow. A new method for mobile robot arm blind grasping using ultrasonic sensors and artificial neural networks. In *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1360–1364, Dec 2013.

[3] A. H. Rajpar, M. U. Keerio, and A. Kawaja. Motion planning of humanoid robot arm for grasping task. In *2007 International Conference on Emerging Technologies*, pages 212–217, Nov 2007.

[4] A. Rastegarpanah, N. Marturi, and R. Stolkin. Autonomous vision-guided bi-manual grasping and manipulation. In *2017 IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*, pages 1–7, March 2017.