北京交通大學

Undergraduate Graduation Design (Thesis)

Study of dual-arm collaboration based on image recognition

Faculty: School of Mechanical and Electronic Control Engineering

Major: Mechanical and Electronic Engineering (Sino-foreign cooperation)

Student Name: Zhenghong Xiao

Student Number: 17222050

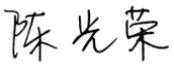
Instructor: Guangrong Chen

Beijing Jiaotong University

Authorization to use copyright of bachelor's thesis

The author of this bachelor's thesis is fully aware of the regulations of Beijing Jiaotong University regarding the retention and use of the bachelor's thesis. The author hereby authorizes Beijing Jiaotong University to compile all or part of the bachelor's thesis into relevant databases for searching and providing reading services, and to preserve and compile it by means of photocopying, microfilming or scanning for inspection and borrowing.

(This authorization statement will apply after the declassification of the confidential dissertation)

Signature of Dissertation Author:  Signature of Instructor. 

Signature Date: June 1, 2021 Signature Date: June 1, 2021

Chinese summary

随着现代科技的发展，机器人的智能化程度也在不断提高，用机器人代替人类完成枯燥、重复的工作，可以大大提升生产效率。视觉信息作为重要的感知信息之一，可以为机器人的工作带来更多的可能性，配备视觉系统的机器人可以实现仓库货物搬运、流水线物品分拣、零件装配等功能。

本文基于机器视觉技术和机器人的智能控制，构建了双臂机器人的手眼系统，并通过识别定位和抓取的实验任务验证了该系统的有效性。

首先，研究了机器人的机械结构及其运动学正解和负解，得出了机器人动力学的数学模型，并通过增加抓手和设计抓手与双臂机器人本体装配的连接器，改进了双臂机器人的硬件系统。

接下来，研究了机器人的视觉识别和定位系统。介绍了相机的选择，分析了相机内部参数校准的原理，并通过相机校准程序减少了图像失真，利用开源数据集训练了 SSD 神经网络并用于物体识别和分类，利用提取像素坐标值和像素点深度值的方法对物体进行了定位，最后将整个视觉系统的 视觉信息最终以 ROS 消息的形式集成到 ROS 环境中供后续使用。

然后，研究了机器人的智能控制系统，建立了机器人的参数文件，利用订阅视觉信息的主题来获取物体类型和三维坐标点的信息，并通过 MoveIt! 中的函数接口来抽象出真实环境中的运动空间和碰撞对象，并通过网络通信协议将路径规划的输出信息发送给机器人，使机器人在模拟和真实环境中的运动轨迹 通过网络通信协议将路径规划的输出信息发送给机器人，使机器人在模拟和真实环境中的轨迹一致。

最后，以 RGB-D 摄像机和双臂机器人为硬件基础平台，通过设置双臂倒水的任务场景，对机器人的手眼系统进行了测试，分析了手眼系统的优缺点，并针对系统的不足之处提出了有效的解决方案。

本文共包含 55 对图、6 张表和 51 个参考文献。

关键词：机器视觉；物体分类；定位和抓取；双臂机器人；ROS 智能控制

ABSTRACT

ABSTRACT:

With the development of modern technology, the intelligence of robots is also increasing, and the use of robots to replace human beings to complete boring Visual information as one of the important perceptual information can bring more possibilities for the work of robots, and robots equipped with vision system can realize the efficiency of production. Visual information as one of the important perceptual information can bring more possibilities for the work of robots, and robots equipped with vision system can realize the functions of warehouse goods handling, assembly line item sorting, parts assembly, etc.

In this paper, based on machine vision technology and intelligent control of robots, we build a hand-eye system for a two-armed dual-arm robot and verify the effectiveness of the system by setting experimental tasks for positioning and grasping.

Firstly, the mechanical structure of the robot and its kinematic positive and negative solutions are studied, the mathematical model of robot dynamics is derived, and the hardware system of the dual-arm robot is improved by adding grippers and designing connectors for assembling the grippers and the body The hardware system of the dual-arm robot is improved by adding grippers and designing connectors for assembling the grippers and the body.

Next, the vision recognition and localization system of the robot is investigated. The selection of the camera is introduced, the principle of the internal parameter calibration of the camera is analyzed and the distortion of the image is reduced by the camera calibration procedure, the SSD neural network is trained and used for object recognition and classification using open-source datasets, and the object is localized by extracting the pixel coordinate values and depth values of the pixel points, and finally the visual information of the whole vision system is integrated into the ROS The visual information of the whole vision system is integrated into the ROS environment in the format of ROS messages for subsequent use.

Then, the intelligent control system of the robot is studied, the parameter file of the robot is established, the object type information and 3D coordinate points are obtained by subscribing to the topic of visual information, the motion space and collision objects

in the real environment are The output information of the path planning is sent to the robot through the network communication The output information of the path planning is sent to the robot through the network communication protocol so that the motion trajectory of the robot in the simulation and the real environment is consistent.

Finally, the hand-eye system of the robot is tested by setting the task scenario of pouring water with two arms using the RGB-D camera and the two-arm robot as the hardware base platform, analyzing the advantages and disadvantages of the hand-eye system, and proposing effective solutions for the The system is based on a two-arm robot with the RGB-D camera and the two-arm robot as the hardware base platform, analyzing the advantages and disadvantages of the hand-eye system, and proposing effective solutions for the shortcomings of the system.

In total, 55 pairs of figures, 6 tables and 51 references are presented in this paper.

KEYWORDS: machine vision; object classification; positioning and grasping; dual-arm robot; ROS intelligent control

Table of Contents

Contents

1 Introduction

1.1 Background and significance of the study

With the development of modern technology, the concept of intelligence has permeated all walks of life. The use of industrial robots equipped with intelligent systems to replace manpower for repetitive and highly accurate tasks saves labor expenses and improves the quality and efficiency of product manufacturing. The degree of intelligence of industrial robots has increased with the continuous improvement of computer science and technology, and their functional loads have become more and more abundant. Currently intelligent robots have been widely used in automotive, pharmaceutical, chemical, food, metal manufacturing and other fields[1] . China's State Council has issued important instructions on the future direction of manufacturing, including a focus on smart manufacturing, and has proposed the concept of "Smart Manufacturing 2025".

Machine vision is an extremely important part of the functional modules of industrial robots. The steps of machine vision implementation include image acquisition, image pre-processing and analysis of the images. Image acquisition is done by a camera with image acquisition capabilities, and different types of images are captured depending on the type of sensor the camera has. The computer receives the image and preprocesses the original image, analyzes the processed image information, classifies the object using visual recognition algorithms, and extracts depth image information to obtain 3D analysis results and object type information.[2] This information can be used in industry for parcels. In industry, this information can be used for parcel sorting, quality inspection, dimensional measurement, etc. These functions can effectively solve the problem of high intensity and low efficiency of manual inspection.

Deep learning is an emerging field of artificial intelligence nowadays, and its construction principle is similar to the process of human learning knowledge. How to combine deep learning with machine vision and robotics to optimize target localization, object grasping and other functions is currently a hot research topic in the field of robotics[3] . With such a combination, an industrial robot can perform the specified tasks with faster speed and higher accuracy, and it can also perform more complex tasks. This

shows that the value that industrial robots can create in the future smart manufacturing industry will be further enhanced with the development of the field of artificial intelligence.

Overall, the research on collaborative control of two arms based on image recognition is of great significance to industrial production. Optimizing the system will not only reduce labor expenses and improve efficiency to promote economic development, but also create social value through the impact of the research, highlighting the country's technological level.

1.2 Current status of domestic and international research

1.2.1 Current status of research on two-armed robots

Foreign research on dual-arm robots began early, with researchers in the United States and Japan leading the advancement of the industry with advanced industrial manufacturing and substantial funding support. The next section will introduce the more representative dual-arm robots and vision research based on dual-arm robots at home and abroad.

Developed by ABB, YuMi is the industry's first 14-axis robot to enable human-robot collaboration. Equipped with a flexible gripper, feeding system, precise vision positioning recognition system, Yumi's single arm has a payload of 0.5kg and a horizontal

vertical range of motion of 559mm.[4] .

Figure 1-1 Yumi dual-arm robot Figure 1-2 Kitech dual-arm robot

by Korea Institute of Technology[5] Kitech's two-armed robot is equipped with two

human-like palms and a vision sensing system, allowing it to work with both arms in a variety of complex scenarios, such as putting together blocks with both hands and carrying goods to designated locations.

The Motoman-SDA series two-arm robot developed by Yaskawa Japan is similar in size to a human being. It is a 15-axis two-arm robot consisting of a body with one axis of rotation and a single arm with seven axes, each arm having a payload of 20 kg, and its horizontal and vertical range of motion of the arm is 2590 mm and 1820 mm, respectively.

Figure 1-3 Motoman-SDA

The Baxter two-arm robot from Rethink Robotics has a single arm with seven degrees of freedom and, unlike traditional industrial two-arm robots, its joint assembly is composed of a series of elastic brakes with a spring between the engine, drive and actuator that monitors external forces and provides feedback. Baxter's arms are flexible and force-sensitive, with a CMOS camera at the end of each arm and the optical axis of the camera oriented parallel to the end joint axis.[6] Baxter's arms are flexible and have good force sensing capabilities. Because of the safety of the Baxter robot and its well-developed sensor system, many researchers have conducted functional extension studies based on this robot.

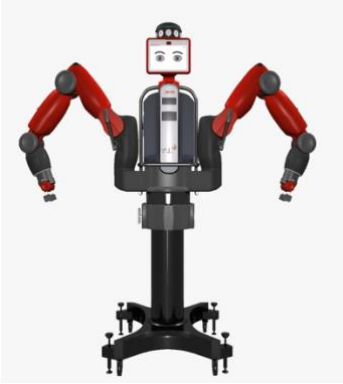


Figure 1-4 Baxter dual-arm robot

The use of two-armed robots is not limited to industry either, as the Rollin Justin

two-armed robot developed by the German Aerospace Center[7] With 51 degrees of freedom throughout its body, it can be used for space operations. Houston Electromechanical Corporation has developed Aquanaut, a two-armed robot for ocean exploration.[8] The robot can be used for remote repair of deep-sea equipment after a deep dive to the mission site. Canadian team researches Dextre two-armed robot[9] can assist astronauts in space walks and can replace astronauts in some dangerous extravehicular operations.

The DSCR5 dual-arm robot developed by Xinsong has a built-in control cabinet with a rich expansion interface, a single arm with a rated load of 5 kg, a single arm working radius of 800 mm, and basic functions such as traction teaching.[10] It is also a physical robot used in this research project.

Figure 1-5 Xinsong dual-arm robot

The police robot developed by Beijing Institute of Aerospace MEMS Technology, the robot integrates a variety of sensor devices, equipped with advanced dual-arm collaboration technology, which can achieve low-latency bionic operation under non-visual conditions, the dual-arm robot can use two arms to collaborate operation, with real-time voice calls, high-definition video playback and the collection of environmental information through sensors at the scene, the operator can control in a safe position The robot can be operated from a safe position to complete rescue work[11] The operator can control the robot from a safe location to complete the rescue work.

From the point of view of the robot hardware system, the form of domestic and foreign two-armed robots is very similar, they have 14 degrees of freedom in both arms, according to the different task scenarios will add degrees of freedom in the legs and waist. Most of the robots integrate a large number of force, torque, vision and other sensors. Through comparative analysis, it can be seen that domestic and foreign dual-arm robots are very similar in industrial task scenarios, but in scenarios such as space operations and

marine operations, China's research in this area is still relatively small.

1.2.2 Current status of machine vision research

French computer scientist Likun Yang's proposed convolutional neural network is seen as a major leap forward in the field of vision[12] He claims that the research was inspired by Kunihiko Fukushima's proposal of a neurocognitive machine[13] . Another big leap in visual recognition was the emergence of ImageNet, in which Professor Feifei Li of Princeton University in the field of computer vision built an open-source database of images based on nouns in the word network, so that each noun contains a large number of images of the thing it represents, based on the problem that the field of visual recognition requires a large number of, multi-category images.[14] .

Pinto at the Robotics Institute of Carnegie Mellon University proposed a CNN-based self-supervised learning algorithm for 3D depth sensor analytical inference grasping with the disadvantages of fitting a difficult model and ignoring the density and mass of the object itself, this model can learn and predict to grasp unknown objects through repeated experiments, after self-built training datasets and using pre-trained networks, finally achieving The two-armed robot achieved a 73% grasping success rate under different conditions[15] Pinto hopes to apply this network to extend object 3D information for non-planar grasping in the future.

Shingo Kitagawa in the JSK lab at the University of Tokyo conducted a visual grasping study using a Baxter robot. Shingo proposed a multi-stage learning method based on CNN for dual-arm grasping of goods for warehouse picking, and the Baxter robot integrated with the software system of the method performed automatic labeling and grasping trials in the real world in a very short period of time A high grasping success rate was achieved. Finally, after tuning the network to obtain a high success rate of 76.7% in 90 trials[16] Shingo suggested that in the future, he would like to equip more dexterous robotic hands to the two-armed robot to achieve more robotic manipulation tasks.

Jeffrey Mahler of Berkeley, California, conducted research on visual grasping experiments based on ABB's YuMi robot. Jeffrey developed a graspable quality convolutional neural network (GQ-CNN) architecture that predicts the robustness of point clouds and trained it on Dex-Net 2.0. After more than 1,000 physical experiments to evaluate the training, the grasping system demonstrated 99% accuracy in 40 location-

based object grasping experiments, and the method was three times faster than point cloud-based methods.[17] Jeffery says that in future work he hopes to achieve 100% success on known objects by using active learning to adaptively acquire grasping points using a GQ-CNN initialization strategy, and he also plans to extend the method to use point clouds from multiple perspectives to accomplish grasping tasks with sequential structure.

Ashvin Nair of Berkeley, California, combines the DDPG algorithm through several algorithms[18] and demonstrations together, using demonstrations to simplify the learning process of the robot. This method can be applied to the robot's learning of any continuous task. After simulation experiments, Nair's robot trained by this method can stack three squares without re-presentation, and if the parameters are adjusted and demonstrated, the robot can stack up to six squares. nair says the main limitation of the method is the sample efficiency when solving difficult tasks.

Guo Di from Tsinghua University proposed and implemented a robot grasping planning method based on a three-dimensional model of the target object, and applied a deep learning method to the grasping task of the robot, using a grasping rectangular box to represent the grasping position of the robot's dexterous hand, and proposed an end-to-end grasping detection convolutional neural network [19] .

Ni Hepeng et al. from Shandong University proposed an image de-duplication algorithm based on time and workpiece position, which was tested on a Delta robot to solve the problem of repeated shots of workpieces by the vision system during the sorting process. The authors obtained the fastest sorting speed of 110 times/min in the solid robot experiment, and obtained a false capture rate of less than 2% missed capture rate of 0[20] The experimental results demonstrate the high stability and accuracy of the algorithm used by the authors.

Chuan-Peng Li of North Central University proposed an improved SSD algorithm (FP-SSD) for the problem that SSD is not effective in locating small objects. mAP of this method reached 79.0%, which improved the accuracy by 1.7% compared with the SSD method[21] .

Regarding visual localization, Dong et al. used Kinect to acquire real-time images of tomatoes in the farm, and after segmenting and de-noising the original images. Using a point cloud library to reconstruct the tomato model in 3D and obtain its 3D coordinate system, experiments show that this system has high accuracy in target localization, and they use this image information combined with a robot with vacuum suction cups to

perform automatic tomato picking operations in the farm [22] They used this image information combined with a robot with vacuum suction cups to automatically pick tomatoes on farms.

In a similar study, Figueroa et al. used a depth camera to capture depth information in color images and scenes. This vision system uses fused data from color cameras and depth sensors to segment objects by distance, use scale-invariant features to describe and identify objects, and and internal parameters of the camera combined with depth information to locate objects relative to the camera's viewpoint. The system is implemented with a robotic arm to locate objects for grasping[23] .

Ma Songhui et al. designed an object detection and localization algorithm based on Mask RCNN and stereo vision in order to improve the picking speed and accuracy of the robot, and completed the autonomous detection and 3D spatial localization of the target to be detected. For the problems that the detection accuracy of neural network may be low and the object contour centroid estimation is inaccurate, ORB descriptors are used to confirm the target contour matching centroid. The experimental results show that the algorithm proposed by Ma Songhui can accurately complete object detection and localization, which is important for the research of fully automated picking robots[24] .

Gené-Mola, Jordi et al. constructed a remote fruit detection system capable of identifying and locating fruits in three dimensions. Most current fruit detection systems are based on two-dimensional image analysis. They used Mask R-CNN instance segmentation neural network for 2D fruit detection and segmentation, generated 3D point clouds of detected apples using motion structure photogrammetry, projected 2D image detection into 3D space, and used a trained support vector machine to remove false positives. This method was tested on 11 Fuji apple trees, which contained a total of 1455 apples. The results showed that by combining instance segmentation with SFM, the system performance improved from an F1 score of 0.816 (2D fruit detection) to 0.881 (3D fruit detection and localization) in[25] .

1.3 Main work of this paper

The subject entities of this paper are the Astra Pro depth camera and the DSCR5 dual-arm robot developed by Sinsung. The key hardware included in this system are the upper computer computing system, the camera with RGB-D sensor, the flexible gripper,

and the dual-arm robot body. The overall schematic diagram of the system arrangement is shown in Figure 1-6.

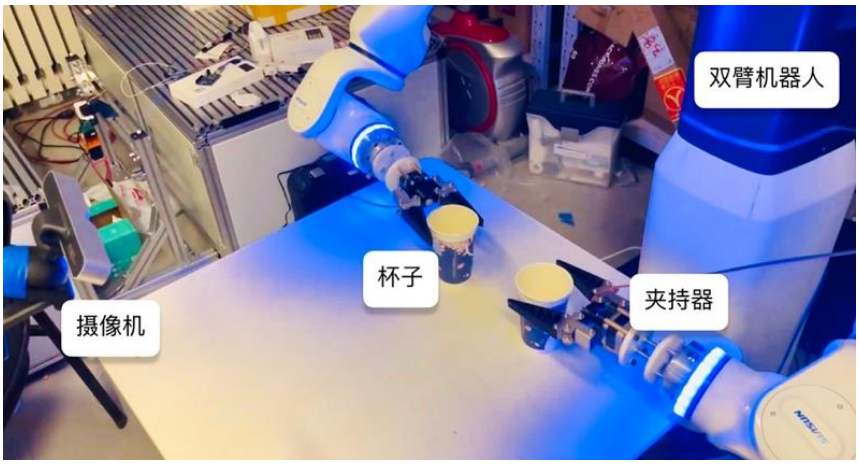


Figure 1-6 Dual-arm robot hand-eye system

Based on the above background, this project will investigate the collaborative system of two-arm robot based on image recognition, and the research areas include the analysis of the robot mechanical structure and its motion and dynamics model, the calibration of the internal and external parameters of the camera, the target detection based on deep learning, the extraction of 3D coordinate points, the collaborative trajectory planning of the two arms and the intelligent control of the robot, and the combination of these modules to complete the water pouring task to verify the The effectiveness of each module is verified.

The specific work on the subject is as follows.

(1) Robot mechanical structure and mathematical model

Solve the D-H table of the robot's single arm, mathematically derive the robot's forward and reverse kinematic solution, dynamics model, analyze the mechanical mechanism of the robot's double arm and its gripper, design the connector independently, and improve the robot hardware system.

(2) Visual perception and positioning

Analyze the theoretical principle of camera parameter calibration and use the code to implement the internal parameter calibration of RGB-D camera, obtain the corrected RGB image and depth image, use the trained neural network SSD (single shot detection) to identify the grasped object and extract the 3D coordinate information of the identified object. The 3D coordinate information of the object and its kind of information are sent to the upper computer system for robot arm path planning.

(3) Two-arm collaboration and intelligent control

Build the ROS-supported URDF robot model format according to the parameters of the real robot, and use the MoveIt plugin to export the SRDF model that can be used for path planning.

Complete the hand-eye calibration to find the external parameters of the camera, use the coordinate transformation package tf in ROS to transform the coordinate points of the object under the camera coordinate system to the base coordinate system of the robot, add the obstacles in the real space after abstraction in the control program, set the dual-arm robot workspace, and use the robot end gripper to reach the gripping point near the 3D coordinates of the object as the task goal to complete the dual-arm robot The path planning is completed and the planning information is output. The ROS output robot arm path planning information is sent to the robot via TCP/IP in robot script language format to realize the control of the physical robot's dual arms. Design experimental scenarios including single-arm experiment, clamping experiment, and dual-arm collaboration experiment to verify the correctness and effectiveness of the intelligent control system of the dual-arm robot.

1.4 Content arrangement and structure of the whole text

Chapter 1 introduces the research background and significance of this paper, including the dual-arm robot, the domestic and international research progress in the field of machine vision, an introduction to the research object of this paper, the completed work and the arrangement of the full text.

Chapter 2 analyzes the mechanical mechanism and kinematics of the robot, derives and verifies the forward and inverse kinematic solutions of the robot, obtains the DH parameters of the robot, derives and verifies the robot dynamics model, designs and prints connectors using 3D modeling software, and refines the robot hardware system.

Chapter 3 investigates the visual perception and recognition localization of the vision system. In this paper, RGB-D cameras are selected as the collectors of visual information, and the internal parameter matrices of RGB and IR cameras are obtained using the Zhang Zhengyou calibration method, respectively, and the SSD neural network is trained with the COCO training set for object recognition and classification using the TensorFlow deep learning framework. The coordinate points of the object in the pixel coordinate system and the depth values are collected, and the 3D coordinate points of the

object in the camera coordinate system are calculated using the tf tool and encapsulated as a topic (Topic) of the ROS system for subsequent use. The causes of errors in visual localization are tested and analyzed, and effective solutions are proposed.

Chapter 4 establishes the simulation model of the robot with the parameters of the real robot, uses the object type and coordinate point information as the input of the intelligent control system, introduces the principle of hand-eye calibration and uses code to implement it to get the parameters outside the camera, uses the tf toolkit of ROS to complete the transformation between multiple coordinate systems within this system, and outputs the 3D coordinate points of the object under test in the base coordinate system of the robot. Call the MoveIt path planning function package in the ROS system, add obstacles such as tables in the robot's real environment, and set the robot's workspace size, and send the planning information to the physical robot to test the consistency of the virtual and real robot movements. Replace the kinematic solver and add a PID controller to optimize the robot's motion planning process.

In Chapter 5, three test scenarios, single-arm experiment, grasping experiment, and two-arm collaborative implementation, are set to experimentally test the correctness and effectiveness of the system and analyze the performance of the hand-eye system.

Chapter 6 concludes the whole text, summarizes the shortcomings of this topic, proposes improvement options and next research directions, and analyzes the social and economic aspects of this system.

2 Robot mechanical structure and mathematical model

A two-armed seven-degree-of-freedom collaborative robot, similar to a human, has seven degrees of freedom in each arm, which allows it to avoid some odd bit patterns in space and to bypass obstacles in the task space relatively easily, thus achieving flexible and accurate motion for each grasping object. Robot kinematics studies the relationship between the robot joint motion variables and their derivatives of each order, and the terminal poses (or robotic jaws) and their derivatives of each order. Compared to traditional industrial robots, it has the ability to optimize the motion and dynamics characteristics, the forces acting on the joints, etc. After completing the simulation modeling, the Matlab tool is used to simulate its workspace and verify the validity of the calculation results.

2.1 Double-arm mechanical structure

2.1.1 Mechanical arm structure

The single arm of the Xinsong dual-arm robot is shown in Figure 2-1, and it has seven degrees of freedom. Unlike traditional six-degree-of-freedom industrial robots, the single-armed Xinsong dual-arm robot with seven degrees of freedom can operate in more complex environments.

Figure 2-1 Xinsong robot seven-degree-of-freedom single arm

The seven joints of this arm are labeled J1, J2, J3, J4, J5, J6, and J7, and it is composed of the following main components.

(1) Base module: The base module is built into the shoulder of the two-arm collaborative robot (J1), and the robot cable is connected to the integrated integrated control cabinet through the base module interface to provide power supply and data

transmission for the robot.

(2) Joint module: The robot is connected by multiple cast aluminum joint modules with built-in drive units (J2~J5).

(3) Wrist joint: The robot is equipped with a wrist joint that integrates J6 and J7 axes.

(4) Electrical system: The electrical system consists of all the electrical components (including drivers, connectors, cables, etc.) that supply and control the motors for each joint.

In particular, the wrist joint is connected using a flange, and the robot ends with a wrist flange with mounting screw holes and pin holes that can be used to mount the end tool. The expansion I/O port on the flange can be used to connect the end-effector structure.

The traditional DH model is built as shown in Figure 2-2. According to the DH model building rules, the coordinate system of each joint is established by building Z_i axis, X_i axis and Y_i axis in turn; then the four parameters of the robot rod's common vertical length a, the angle α between the joint axes, the joint distance d and the robot joint rotation angle θ are determined according to the relative position of the joint coordinate system.[26] .

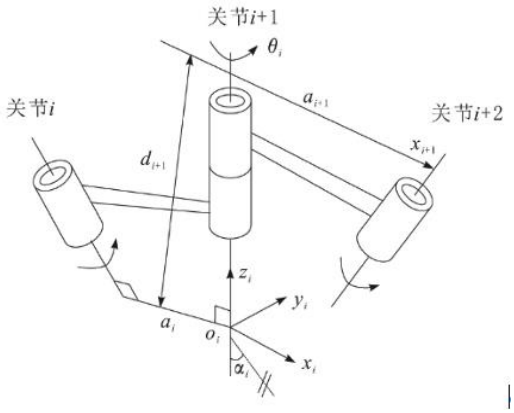


Figure 2-2 Relationship of coordinate system in DH model

The joint coordinate system of the robot is established according to the DH method and schematically shown as follows.

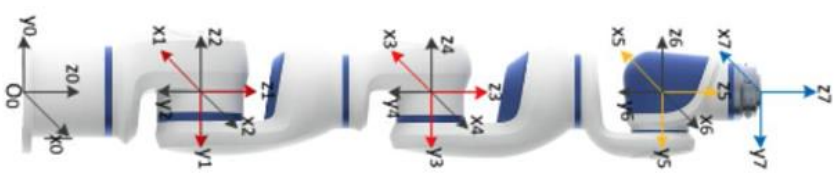


Figure 2-3 Coordinate system of robot arm

The robot's 14 joint coordinate system and base coordinate system are established according to the principle of robot arm coordinate system establishment, as shown in the following figure.

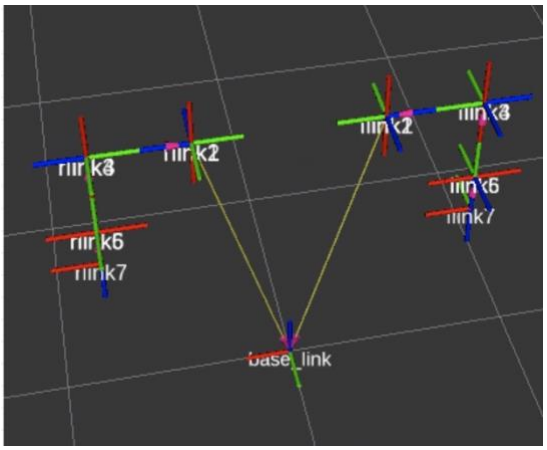


Figure 2-4 Coordinate system of two-arm robot

The basic parameters of the single arm are shown in Table 2-1.

Table 2-1 Joint motion range of two-arm robot

Joint	Range of	Movement
1	-180 to +180	120
2	-105 to +105	120
3	-180 to +180	180
4	-115 to +115	180
5	-180 to +180	180
6	-110 to +110	180
7	-180 to +180	180

2.1.2 Clamp and its connector structure

Since this dual-arm robot does not come with a gripper, so we purchased a flexible gripper for clamping objects, using Raspberry Pi 4B to control the stepper motor on the gripper, through python to write the Raspberry Pi GPIO control program can realize the opening and closing of the gripper, in the actual test the gripper can clamp cups and other objects, the parameters show that the gripper weighs 546g, can clamp the maximum 1.5kg

The parameters show that the gripper weighs 546g and can grip objects of up to 1.5kg.

The gripper is powered by an independent power supply and comes with a stepper motor controller with three interfaces, one of which is a ground port, and the other two ports are given 0/1 or 1/0 to open and close the gripper, and the angle of opening and closing can be controlled by the time of the given level.

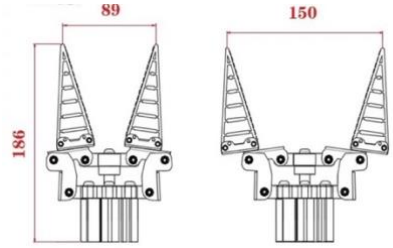


Figure 2-5 Mechanical structure of the gripper and its parameters

The connector is designed through Solidworks 3D modeling to connect the gripper to the end joint. Both the dual-arm robot and the gripper come with four screw holes, and the connector structure is shown below.

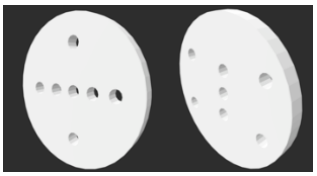


Figure 2-6 Gripper Connector

The two parts of the connector have multiple through-holes, and the connector entity is printed by a 3D printer to connect the gripper, the connector, and the robot body assembly.

Figure 2-7 Assembly diagram

2.2 Robot kinematic model

2.2.1 Positive kinematics

In order to obtain the forward and inverse solutions of the robot kinematics, it is necessary to solve the forward and inverse kinematics using the D-H method and the analytical method, respectively, and verify their correctness using Matlab/Gazebo.

D-H modeling method[26] is a modeling method proposed by Denavit and Hartenberg, mainly used in robot kinematics. This method establishes a coordinate system on each linkage, and transforms the coordinates on both links by a chi-square coordinate transformation, and uses the chi-square change to establish the relationship between the first and end coordinate systems in a system with multiple links in series [27] . Using the DH method to model the system of the robot, the D-H parameters of the robot can be obtained as shown in Table 2-2.

Table 2-2 D-H parameters of robotic arm

Linkage number	Joint angle θ_i	Joint bias d_i	Linkage length a_{i-1}	Connecting rod
1	θ_1	0.310	0	π/2
2	θ_2	0	0	π/2
3	θ_3	0.400	0	π/2
4	θ_4	0	0	π/2
5	θ_5	0.400	0	π/2
6	θ_6	0	0	π/2
7	θ_7	0.175	0	0

Performing the mathematical derivation, the general equation for the linkage change of the robot is

$$
\begin{aligned}
{}_{i-1}^{i}T_i &= T_{RZ\text{-}1}(q_i)T_z(d_i)T_x(a_i)T_{RX}(\alpha_i) \\
&= \begin{bmatrix} cq_i & -sq_i & 0 & 0 \\ sq_i & cq_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_i & -s\alpha_i & 0 \\ 0 & s\alpha_i & c\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} cq_i & -c\alpha_i sq_i & s\alpha_i sq_i & a_i cq_i \\ sq_i & c\alpha_i cq_i & -s\alpha_i cq_i & a_i sq_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{2-1}
$$

The analysis of the positive kinematics of the robot arm is to calculate the positional attitude of the end of the robot given the rotation angle of each joint, where the key problem is to solve the mapping problem between the joint space and the operation space. The expression for finding the end motion poses is as follows.

$$
{}_{7}^{0}T = {}_{1}^{0}T(\theta_1){}_{2}^{1}T(\theta_2){}_{3}^{2}T(\theta_3){}_{4}^{3}T(\theta_4){}_{5}^{4}T(\theta_5){}_{6}^{5}T(\theta_6){}_{7}^{6}T(\theta_7) \tag{2-2}
$$

This result is expressed analytically as

$$
{}_{7}^{0}T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2-3}
$$

This matrix represents the correspondence between the position of the end-effector center position based on the base coordinates and the attitude.

The results of the robot toolbox computational simulation simulations and the derived computational results are compared separately to verify the correctness of the positive kinematic solution. As shown in the figure below, the robot's position *XYZ* is (0.975, 0.000, 0.310) and the pose is represented as *RPY* (0.000, 90.000, 180.000) showing that the output results of the positive kinematic analytical solution assignment match exactly with the simulation calculation results in the toolbox, proving that the analytical solution results are valid.

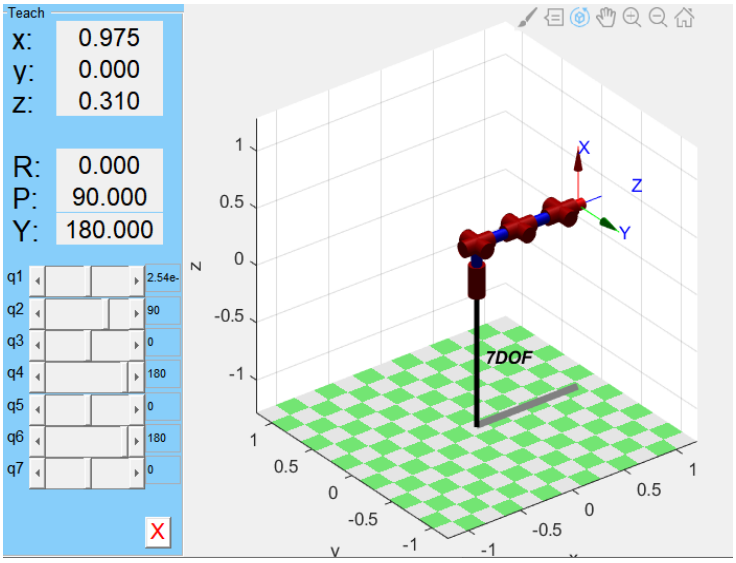


Figure 2-8 Positive kinematics robot end position schematic

2.2.2 Inverse kinematics

The inverse kinematics of the robot is based on the known end position of the robot arm, and the commonly used methods are numerical and analytical methods. However, the analytical method is relatively small and suitable for real-time control. To get a useful inverse solution, an additional joint rotation angle should be added. The idea is to solve $\theta_{1,}\theta_{2,}\theta_4$ based on the end attitude and joint angle θ_3 , followed by $\theta_{5,}\theta_{6,}\theta_7$ using the end attitude, which is derived as follows[28] The derivation is as follows

Assume that the positional matrix of the end center of the robot arm is known to be

$$\theta_4 = \pm\left(\pi - \arccos\left(\frac{p_x^2 + p_y^2 + p_z^2 - d_1^2 - d_2^2}{2d_1 d_2}\right)\right) \tag{2-4}$$

where s_1 and c_1 represent $\sin\theta_1$、$\cos\theta_1$ respectively. Calculating $p_x^2 + p_y^2 + p_z^2$ yields.

$$\theta_4 = \pm\left(\pi - \arccos\left(\frac{p_x^2 + p_y^2 + p_z^2 - d_1^2 - d_2^2}{2d_1 d_2}\right)\right) \tag{2-5}$$

After finding θ_4 and based on p_x and the previously assigned θ_3 , find θ_2 :

$$\theta_2 = \arcsin\left(\frac{p_z}{\sqrt{(-d_1 s_4 c_3)^2 + (d_1 + d_2 c_4)^2}} - \arctan\left(\frac{d_1 + d_2\cos\theta_4}{-d_2\sin\theta_4\cos\theta_3}\right)\right) \tag{2-6}$$

According to the analytical solution of p_x ,p_y contains both θ_1 , while θ_2 ,θ_3 ,θ_4

have been solved, again brought into the original equation can be obtained

$$\theta_1 = \arctan\left(\frac{(p_x n_y - p_y n_x)(m_y n_x - m_x n_y)}{(m_x n_y - m_y n_x)(p_x m_y - p_y m_x)}\right)$$
$$= \arctan\left(-\frac{(p_x n_y - p_y n_x)}{(p_x m_y - p_y m_x)}\right) \quad (2\text{-}7)$$

Eq.

$$m_x = -d_2 s_4 c_3$$
$$n_x = d_2 s_4 c_2 c_3 + (d_1 + d_2 c_4) s_2$$
$$m_y = d_2 s_4 c_2 c_3 + (d_1 + d_2 c_4) s_2$$
$$n_y = d_1 s_4 s_3$$

5, 6, and 7 joint angles are used to determine the end motion attitude of the robot arm, using the conditions ${}_0^7T$, and ${}_4^0T = {}_1^0T(\theta_1)\,{}_2^1T(\theta_2)\,{}_3^2T(\theta_3)\,{}_4^3T(\theta_4)$ can be found at ${}_4^0T$, and.

$${}_7^4T = ({}_4^0T)\,{}_7^{-1}T = \begin{bmatrix} n_{11} & n_{12} & n_{13} & p_x \\ n_{21} & n_{22} & n_{23} & p_y \\ n_{31} & n_{32} & n_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2\text{-}8)$$

From this, we can find θ_5、 θ_6、 θ_7 as follows

$$\begin{cases} \theta_5 = \arctan 2(n_{33}, n_{13}) \\ \theta_6 = \arccos(-n_{23}) \\ \theta_7 = \arctan 2(n_{22}, n_{21}) \end{cases} \quad (2\text{-}9)$$

The end-joint angle of each joint is calculated by importing the end-jaw posture matrix obtained previously as a known quantity into the analytical solution formula, and referring to the derived analytical solution, the end-joint angles are as follows

$$\theta = [0 \quad pi/2 \quad pi/2 \quad pi \quad pi \quad pi \quad 0]$$

The inverse solution is performed and the result of Q1 is found as

[-0.0000 1.5708 -0.6387 3.1416 0.0000 3.1416 -0.9320]

A comparative validation of the calculated results is shown below.

$$T = \begin{bmatrix} 0 & 0 & 1 & 0.975 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0.31 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

As can be seen in Figure 2-9, the left figure shows the end poses displayed by inputting each joint angle Q1 (radian system), and the right figure shows the results after plotting the end poses matrix, where the positions *XYZ*, correspond to the poses *RPY* results, verifying the accuracy of the kinematic algorithm and proving that the algorithm is effective.

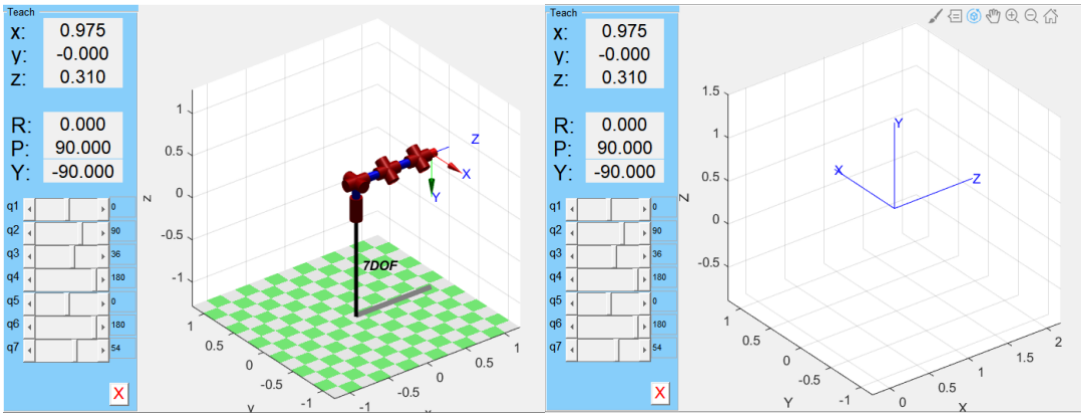


Figure 2-9 Comparison of simulation results verification

2.2.3 Robot Workspace Analysis

Therefore, the flexibility of the robot can be verified based on the intersecting volumes as an evaluation index in the case of tightly coordinated movements. When evaluating the flexibility criteria, it is important to look at the consistency of the operation of the same target, e.g., the ability to move the object smoothly in any direction in the space, rotation, etc.[29] . When the point A is the center of mass of the target object, the surrounding points can be studied according to the kinematic relationship between the two points within the workpiece, and then the degree of coordinated operation of the left and right arms can be analyzed. *J* represents the Jacobi matrix, and the area enclosed by the red line represents the intersecting volume of the operable ellipsoid, as shown in Figure 2-10. Regarding the calculation of the spatial volume of the intersecting ellipsoid can be referred to TOMM[30] method for the approximate metric calculation.

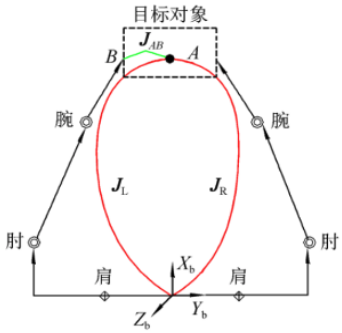


Figure 2-10 Coordinated execution of tasks by both arms

After calculating the above Jacobi matrix, D-H parameters, and positive and negative kinematics solutions, the robot toolbox was applied again to re-build the robot arm model of the two-arm robot and set the corresponding robot parameters as shown in Figure 2-11.

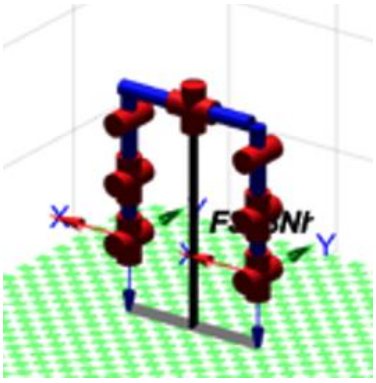


Figure 2-11 Matlab model of a two-armed robot

The simulation was performed for the workspace of this two-armed robot, and the simulation results were obtained as shown in Figure 2-12.

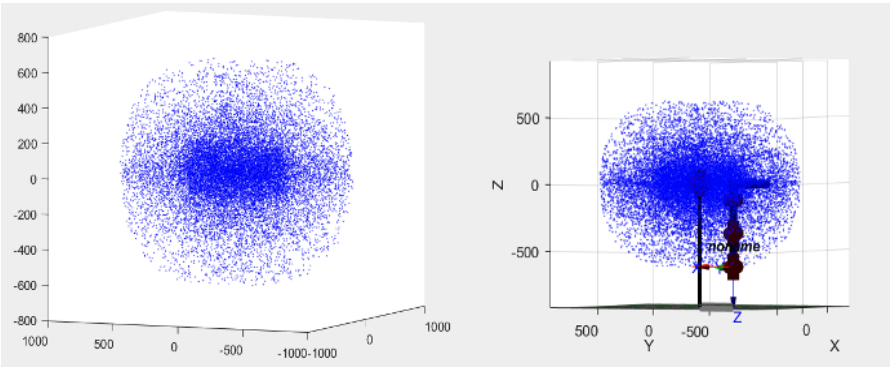


Figure 2-12 Simulation of a two-armed robot workspace

Due to the limitation of computation, the simulation points are set to 20,000 simulation points. From the figure, it can be seen that the distribution of the simulation points is relatively dense near the center region of both arms, showing an almost

ellipsoidal shape.

After that the simulation points in each space are normalized with the volume of the ellipsoid by the formula:

$$\widehat{V}_i = \frac{V_i}{\max(V_1, V_2, ..., V_n)} \tag{2-10}$$

The resulting $\widehat{V}_i$ represents the ellipsoidal volume of the intersection operability at point i after normalization process, and V_i represents the ellipsoidal volume of the intersection operability at point i. The geometric topological relationship between a single and the total. Using the visualization process, we continue to program the graphical solution of the distribution of the flexible operation area of the two-armed machine heat in space, and use the chromatographic way to indicate the degree of flexible operation, whose color is closer to the red end, representing the better operability the higher flexibility, the specific results are shown in Figure 2-13 below.

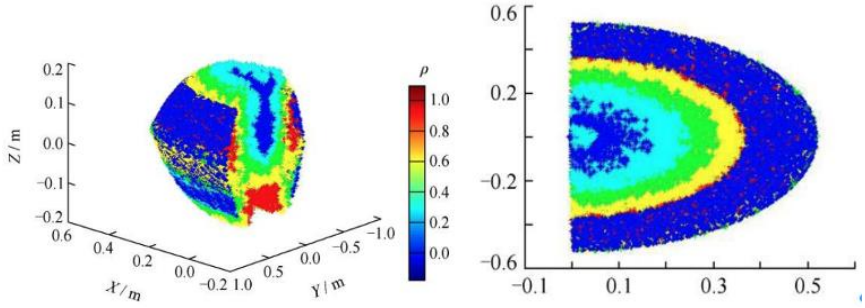


Figure 2-13 Schematic diagram of flexibility analysis in ellipsoidal space

From the figure, it can be analyzed that flexibility, as an indicator of smooth robot operation, shows a certain distribution pattern, with the robot body constrained at the center of the robot, so flexibility is poor. Looking outward to the center of the entire collaborative space, the color becomes red, indicating that the flexibility becomes better. Overall, the robot in the overlapping area of the workspace, is from the outer edge toward the central area is the process of gradually getting better flexibility, which also verifies the correctness of the previous hypothesis.

2.3 Robot dynamics model

The problem studied in dynamics is the relationship between the forces acting and the motion. The corresponding mathematical model is first established, and then the structural characteristics of the robotic arm, which is nonlinear and has complex coupling

relationships, are modeled and analyzed by simulation. After completing the results of the robot arm dynamics analysis, the joints are analyzed for load bearing and force changes on the robot arm, etc., so that the motion can be changed for subsequent optimization to equalize the forces on the joints. The kinetic equations can be used to determine the relationship between the forces and moments applied to the corresponding joints, the robot acceleration, the rotational inertia and the angular acceleration[31] .

Robot dynamics research is based on the idea of solving force problems recursively from back to front, or from front to back. The corresponding research methods are both the Newton-Euler method based on force balance and the Lagrangian method based on energy balance. The relationship between the forces required to cause motion, i.e., robot motion, inertia, and moments, is studied.

Kinetic positive problem: find the motion response with known joint moments.

Kinetic inverse problem: find the joint moment for a known motion.

As shown in Figure 2-14 below, the robotic arm is simplified into a two-link mechanism, which is modeled and analyzed. Lagrange's equation is used to establish the dynamics equation.

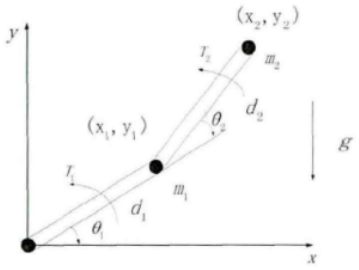


Figure 2-14 Schematic diagram of the model of the two-linked rod

For the center of mass set to m_1, m_2 and the length of the connecting rod set to $d_1, d_2, \theta_1, \theta_2$ is the angle of rotation of the connecting rod. Then calculate the kinetic energy K_1 and the potential energy P_1 of the first connecting rod.

$$K_1 = \frac{1}{2} m_1 d_1^2 \theta_1^2 \tag{2-11}$$

$$P_1 = m_1 g d_1 \cos\theta_1 \tag{2-12}$$

Kinetic energy of the second linkage K_2 , potential energy P_2 :

$$K_2 = \frac{1}{2} m_2 v_2^2 \tag{2-13}$$

$$P_1 = m_1 g d_1 \cos\theta_1 \tag{2-14}$$

And the relationship between velocity and position regarding the connecting rod is

$$v_2^2 = \dot{x}_2^2 + \dot{y}_2^2$$
$$x_2 = d_1\cos\theta_1 + d_2\cos(\theta_1+\theta_2)$$
$$\dot{x}_2 = -d_1\dot{\theta}\sin\theta_1 - d_2\left(\dot{\theta}_1+\dot{\theta}_2\right)\sin(\theta_1+\theta_2)$$
$$y_2 = d_1\sin\theta_1 + d_2\sin(\theta_1+\theta_2)$$
$$\dot{y}_2 = d_1\dot{\theta}_1\cos\theta_1 - d_2\left(\dot{\theta}_1+\dot{\theta}_2\right)\cos(\theta_1+\theta_2)$$

The kinetic potential energy equation for reintegrating the second linkage is

$$K_2 = \frac{1}{2}m_2 d_1^2\dot{\theta}_1^2 + \frac{1}{2}m_2 d_2^2\left(\dot{\theta}_1+\dot{\theta}_2\right)^2 + m_2 d_1 d_2\left(\dot{\theta}_1^2+\dot{\theta}_1\dot{\theta}_2\right)\cos\theta_2$$
$$P_2 = m_2 g[d_1\sin\theta_1 + d_2\sin(\theta_1+\theta_2)] \tag{2-15}$$

The expression of Lagrange's equation is given by

$$T_i = \frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i}, i = 1, 2, ..., n \tag{2-16}$$

So, the equation q is representing the coordinates, $\dot{q}$ is representing the corresponding velocity and T_i is the force matrix.

$$L = K - P \quad K = K1 + K2 \quad P = P1 + P2$$

Therefore, finding the partial derivative and inverse of L yields

$$\frac{\partial L}{\partial \dot{\theta}_1} = (m_1+m_2)d_1^2\dot{\theta}_i + m_2 d_1^2\left(\dot{\theta}_1+\dot{\theta}_2\right) + m_2 d_1 d_2\left(2\dot{\theta}_1+\dot{\theta}_2\right)\cos\theta_2$$
$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} = (m_1+m_2)d_1^2\ddot{\theta}_i + m_2 d_1^2\left(\ddot{\theta}_1+\ddot{\theta}_2\right) + m_2 d_1 d_2\left(2\ddot{\theta}_1+\theta_2\right)\cos\theta_2 - m_2 d_1 d_2\left(2\dot{\theta}_1\dot{\theta}_2+\dot{\theta}_2^2\right)\sin\theta_2$$
$$\frac{\partial L}{\partial \theta_1} = -(m_1+m_2)gd_1\cos\theta_1 - m_2 g d_2\cos(\theta_1+\theta_2)$$
$$\frac{\partial L}{\partial \dot{\theta}_2} = m_2 d_2^2\left(\dot{\theta}_1+\dot{\theta}_2\right) + m_2 d_1 d_2\dot{\theta}_1\cos\theta_2$$
$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}_2} = m_2 d_2^2\left(\ddot{\theta}_1+\ddot{\theta}_2\right) + m_2 d_1 d_2\cos\theta_2 - m_2 d_1 d_2\dot{\theta}_1\dot{\theta}_2\sin\theta_2$$
$$\frac{\partial L}{\partial \theta_2} = -m_2 d_1 d_2\left(\dot{\theta}_1^2+\dot{\theta}_1\dot{\theta}_2\right)\sin\theta_2 - m_2 g d_2\cos(\theta_1+\theta_2)$$

Kinetic equations for the moments on the robot arm T_1 and . T_2

$$T_1 = [(m_1+m_2)d_1^2 + m_2 d_2^2 + 2m_2 d_1 d_2\cos\theta_2]\theta_1 + [m_2 d_2^2 + m_2 d_1 d_2\cos\theta_1]\theta_2$$
$$-m_2 d_1 d_2\left(2\dot{\theta}_1\dot{\theta}_2+\dot{\theta}_2^2\right)\sin\theta_2 + (m_1+m_2)gd_1\cos\theta_1 + m_2 g d_2\cos(\theta_1+\theta_2) \tag{2-17}$$
$$T_2 = [m_2 d_2^2 + m_2 d_1 d_2\cos\theta_2]\ddot{\theta}_1 + m_2 d_2^2\ddot{\theta}_2 + m_2 d_1 d_2\dot{\theta}_1^2\sin\theta_1 + m_2 g d_2\cos(\theta_1+\theta_2)$$

By a similar derivation, expressions for the total kinetic energy and total potential

energy of the robot can be obtained, and the Lagrangian function for the two-armed robot system is found as

$$\begin{aligned} L &= K - P \\ &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=1}^{i}\mathrm{Trace}\left(\frac{\partial T_i}{\partial \theta_j} I_i \frac{\partial T_i^T}{\partial \theta_k}\right)\dot{\theta}_j\dot{\theta}_k + \frac{1}{2}\sum_{i=1}^{n} I_{ai}\dot{\theta}_i^2 + \sum_{i=1}^{n} m_i g^T T_i^i r_i \end{aligned} \tag{2-18}$$

The kinetic equations of the system are

$$\begin{aligned} F_i &= \frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}_p} - \frac{\partial L}{\partial \theta_p} \\ &= \sum_{j=1}^{n}\sum_{k=1}^{j}\mathrm{Trace}\left(\frac{\partial T_j}{\partial \theta_k} I_j \frac{\partial T_j^T}{\partial \theta_i}\right)\ddot{\theta}_k + I_{ai}\ddot{\theta}_i \\ &+ \sum_{j=1}^{n}\sum_{k=1}^{j}\sum_{m=1}^{j}\mathrm{Trace}\left(\frac{\partial^2 T_i}{\partial \theta_k \partial \theta_m} I_j \frac{\partial T_j^T}{\partial \theta_i}\right)\dot{\theta}_k\dot{\theta}_m \\ &- \sum_{i=1}^{n} m_j g^T \frac{\partial T_{i_i}}{\partial \theta_i} r_i \\ &= \sum_{j=1}^{n} D_{ij}\ddot{\theta}_j + I_{ai}\ddot{\theta}_i + \sum_{j=1}^{n}\sum_{k=1}^{n} D_{ij}\dot{\theta}_j\dot{\theta}_k + D_i \end{aligned} \tag{2-19}$$

The inertia is expressed in the equation as

$$D_{ij} = \sum_{p=\max i,j}^{6}\mathrm{Trace}\left(\frac{\partial T_p}{\partial \theta_j} I_p \frac{\partial T_p^T}{\partial \theta_i}\right)$$

The coefficient of centripetal acceleration is

$$D_{ijk} = \sum_{p=\max i,j,k}^{6}\mathrm{Trace}\left(\frac{\partial^2 T_p}{\partial \theta_j \partial \theta_k} I_p \frac{\partial T_p^T}{\partial \theta_i}\right)$$

The gravitational force is expressed as

$$D_i = \sum_{p=i}^{6}\left(-m_p g^T \frac{\partial T_{p_p}}{\partial \theta_i} r_p\right)$$

The Lagrangian method is a relatively simple method for solving the dynamical equations of a system as opposed to other relatively simple methods such as the Newtonian Eulerian method, where the driving moment of each joint is associated with the derived dynamical equations.

2.4 Summary of this chapter

This chapter analyzes the mechanical structure of the robot's arm and its gripper, designs and prints connectors for connecting the robot to the purchased gripper using a 3D modeling tool, and introduces the Raspberry Pi into the control system for controlling the opening and closing of the two grippers. The kinematic model of the robot was derived by mathematically deriving the forward and reverse kinematic solutions of the robot and bringing the results into the simulation environment to verify their correctness.

3 Visual perception and recognition localization

The vision system in this paper runs on Ubuntu 16.04 and ROS kinetic, and the object recognition part of the deep learning neural network program is based on the framework Tensorflow 1.14.0.

3.1 Introduction to the program dependency environment

Ubuntu is currently the most popular distribution of Linux desktop system, and is loved by developers for its stability and ease of use. The Ubuntu project is openly committed to the principles of open source software development, encouraging people to use free software, study how it works and develop improvements.

Figure 3-1 Operating Environment

ROS system is a loosely coupled distributed architecture based on publish/subscribe mechanism, which can be used to achieve intelligent control of various robotic drones. ROS is widely used in robotics systems for its power and ease of use.

ROS contains functional modules such as messaging, recording and playback of messages, and function packages.[32] . The following is a description of its main functions.

(1) Message transmission

The ROS control system is composed of a total number of nodes with different functions. Nodes can communicate with each other through ROS predefined message format or user-defined message format. Currently, ROS supports three types of communication, topic for one-way message sending/receiving, service for two-way message requesting/responding, and action for two-way message goal/result/feedback.

(2) Recording and playback of information

Data is recorded and played back through 'rosbag', a feature that not only avoids reusing cadaver robots for algorithm development and validation, but also ensures consistency of data from one experiment to the next.

(3) Function package

ROS has a large number of callable function packages, this article mainly uses MoveIt, Easy_handeye, camera_calibration these packages.

MoveIt is a robotics-related toolset that integrates a variety of SOTA libraries that allow users to configure robot UDF files for simulation or physical robotics including: motion planning, manipulation, 3D sensing, kinematics, collision detection, control, and navigation.

Easy_handeye is a hand-eye calibration method provided by ROS, which provides us with a set of visualization tools to complete the external reference calibration for both eye-in-hand and eye-out-of-hand scenarios.

Camera_calibration is a camera calibration function package based on Zhang Zhengyou's calibration method, which can get the camera's internal parameter matrix, aberration coefficient, etc. through the algorithm.

ROS supports writing a launch file to start multiple nodes at once, or you can preset parameters in the launch file and read them in the program, or you can use the rosrun command to start a single node. string, etc.

TensorFlow is a deep learning framework developed by Google that supports computing using CPUs or GPUs.[33] TensorFlow supports multiple platforms and it is widely used for programming implementations of various machine learning algorithms.

3.2 Camera selection

The cameras commonly used in industry contain monocular cameras, binocular cameras, and cameras with structured light ranging capabilities, and since this experiment requires more accurate 3D point cloud information, the literature[34] As this experiment requires more accurate 3D point cloud information, the literature points out that monocular cameras have very limited ability to obtain depth information, and the target object itself is thicker when its positioning error is large, while binocular cameras are generally more expensive. Therefore, this paper will use the Astra Pro camera with structured light ranging to capture image information and distance information.

Figure 3-2 Astra Pro Camera

This camera has two microphones, a face perception sensor, an infrared projection module, an infrared image acquisition module and a color image acquisition module. The resolution of the color image is 640*480, the frames per second (FPS) is 30, the depth image resolution is up to 1280*1024, the FPS is 7. The theoretical positioning error is plus or minus 3mm, the field of view of the depth information is 58.4 degrees horizontally and 45.5 degrees vertically. The data transmission delay is 30-45 milliseconds, and it supports Windows, Linux system for development and use.

Depth ranging techniques can be divided into passive ranging techniques and active ranging techniques. Passive ranging technology means that the sensor receives the light emitted or reflected from the scene and forms an image.[35] The typical passive ranging technique is binocular vision ranging. Active ranging techniques are hardware systems

that rely on their own light emitting receiver modules to emit light and receive it after reflection, and record this time for distance calculation to obtain a depth image. Common active ranging techniques include TOF ranging technique[36] Phase laser ranging technique[37] and structured light ranging techniques[38] .

The camera sends light pulses to the object and uses the sensor to receive the light returned from the object, because the speed of propagation of the light signal in the air is a constant value, so as long as the time of flight is known, through the calculation can be derived from the corresponding target distance, TOF range measurement will use this principle to measure the distance of the target entity[39] The principle can be expressed by the following equation.

$$d = \frac{1}{2}\left(n*\lambda + \frac{\varphi}{360}*\lambda\right) \qquad (3\text{-}1)$$

Where λ is the wavelength of the signal, n is the number of wavelengths experienced by the signal during the flight time, andφ is the phase of the signal when it returns.

Phase laser distance measurement is accurate to the micron level, using a modulated signal to modulate the light intensity and measure the phase difference to indirectly measure the time and calculate the distance value to achieve the distance measurement function.

The Astra Pro camera used in this paper uses structured light ranging technology, which is based on the principle of infrared projection sensors emitting laser scatter over distance through grating diffraction, the infrared camera can capture the laser scatter image in the visible range and form a parallax map, combined with the camera model to calculate the depth value of each pixel[40] The depth value of each pixel can be calculated by combining the camera model.

Figure 3-3 Color and depth image information

3.3 Camera calibration principle and its implementation

The purpose of camera calibration is to obtain the internal and external parameters of the camera in order to reduce distortion and to complete the transformation between the camera coordinate system and the pixel coordinate system. The calibration uses points with known positions in space and their positions in the image to estimate the parameters in the camera model. While the calibration plate of the traditional calibration method is three-dimensional and needs to be very precise and difficult to produce, the method proposed by Professor Zhengyou Zhang[41] Between the traditional calibration method and the self-calibration method, the Zhengyou Zhang calibration can be performed by printing out only a checkerboard grid. It also improves the accuracy and is easy to operate compared to self-calibration. Zhang Zhengyou calibration method is widely used in the field of camera calibration.

The basic principles of calibration are.

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K[\,r_1 \quad r_2 \quad r_3 \quad t\,]\begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = K[\,r_1 \quad r_2 \quad t\,]\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{3-2}$$

Assume that the template plane is in the plane of the world coordinate system $z_c = 0$, where K is the internal parameter matrix of the camera, $[\,X \quad Y \quad 1\,]^T$ is the flush coordinate of the template screen edge, $[\,u \quad v \quad 1\,]^T$ is the flush coordinate of the corresponding point projected on the template plane onto the image plane, and $[\,r_1 \quad r_2 \quad r_3\,]$ and t are the rotation matrix and translation vector of the camera coordinate phase system with respect to the world coordinate system, respectively, and the rotation matrix can be expressed as

$$H = [\,h_1 \quad h_2 \quad h_3\,] = \lambda K[\,r_1 \quad r_2 \quad t\,], r_1 = \frac{1}{\lambda}K^{-1}h_1, r_2 = \frac{1}{\lambda}K^{-1}h_2 \tag{3-3}$$

According to its property we obtain $r_1^T r_2 = 0$ and $\|r_1\| = \|r_2\| = 1$, using this property we can obtain two constraint equations for the camera internal reference lifting.

$$\begin{cases} h_1^T K^{-t} K^{-1} h_2 = 0 \\ h_1^T K^{-t} K^{-1} h_1 = h_2^T K^{-t} K^{-1} h_2 \end{cases} \tag{3-4}$$

In order to solve for the 5 unknown internal parameters of the camera, the number

of images to be captured is greater than or equal to 3[42] .

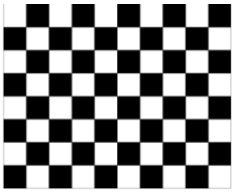
Use GitHub's camera calibration code based on Zhengyou Zhang's calibration method[42] of the camera calibration code, complete the following steps, you can obtain the internal and external parameters of the camera. The specific steps are.

(1) Print a 7*9 calibration plate as shown in Figure 3-4 and paste it onto a flat object, keeping the calibration plane flat. Use calipers to check if the distance between the features of the printed pattern is correct. For the pattern in the calibration plate, the distance between the intersection points of the black and white corners should be exactly 3cm.

(2) Hold the calibration plate, fix the image sensor, move and rotate the plate.

(3) Start the recording program, and then press the space bar to record each image. The calibration pattern (indicated by the color lines overlapping the calibration pattern, as shown in Figure 3-5) should be detected, and the image should be clear and stable.

(4) The calibration plate should be in the calibration plane at all times, and the calibration process should be rotated and calibrated at at least two distances, with 100 images obtained for each calibration set.



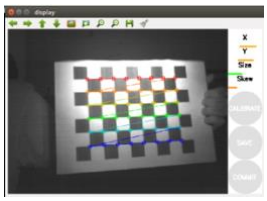


Figure 3-4 Printed Calibration Plate Figure 3-5 Calibration Calibration Pattern

Repeat steps 1-4 twice to complete the calibration of the RGB and IR sensors, after the completion of the calibration program will generate a .yaml file in the specified path, this file is the camera's internal parameters file, the path of the parameters file added to the ROS package under the corresponding launch file after each run the camera image system will get the internal parameters of the calibrated image.

The following results were obtained after calibration.

Table 3-1 Camera calibration results

Parameter Name	RGB Sensors	Infrared sensors
Internal parameter matrix	$\begin{bmatrix} 602 & 0 & 304 \\ 0 & 604 & 265 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 574 & 0 & 315 \\ 0 & 585 & 211 \\ 0 & 0 & 1 \end{bmatrix}$

Distortion factor	$[0.11, -0.32, 0.01, 0.014, 0]$	$[574, 0, 315, 0, 585, 212, 0, 0, 1]$

3.4 Positive and negative changes of the camera model

Robot vision systems usually have the following coordinate systems[43] : The

(1) World coordinate system w : The coordinate system associated with the real world of the shooting.

(2) Camera coordinate system c : A coordinate system fixed on the camera with the origin at the optical center of the camera, the z-axis coinciding with the optical axis, and the x- and y-axes parallel to the coordinate axes of the image plane.

(3) Image plane coordinate system: The coordinate system associated with the virtual image plane. The origin is at the intersection of the optical axis and the image plane.

(4) Image pixel coordinate system i : is the discrete coordinate system with the origin of the coordinate system in the upper left corner of the image.

The surface camera model is to establish the transformation relationship of a known point in space coordinates from world coordinates into image pixel coordinates, such a change is the positive change of the camera model, and the process can be expressed in the equation as

$$p^w \rightarrow p^c \rightarrow q^c \rightarrow q^i \tag{3-5}$$

where p^w is the world coordinate system and p^c is the camera coordinate system, and their transformation relationship can be expressed as follows

$$p^c = {}^cH_w p^w \tag{3-6}$$

where ${}^cH_w$ is the chi-square transformation matrix from the camera coordinate system to the world coordinate system, q^c is the image plane coordinates, and its transformation relationship with p^c can be expressed as

$$q^c = \begin{pmatrix} u \\ v \end{pmatrix} = \frac{f}{c} \begin{pmatrix} x^c \\ y^c \end{pmatrix} \tag{3-7}$$

q^i is the pixel coordinate, which corresponds to the image coordinate as

$$q^i = \begin{pmatrix} r \\ c \end{pmatrix} = \begin{pmatrix} \frac{v}{S_y} + C_y \\ \frac{u}{S_x} + C_x \end{pmatrix} \tag{3-8}$$

where S_x and S_y are the scale factors, respectively, and the interval of pixels on the CCD chip. C_x, C_y are the coordinates of the projection point of the optical center on the image plane. If the lens aberration is not considered the complete camera model can be expressed by the following equation.

$$z^c \begin{bmatrix} c \\ r \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{f}{S_x} & 0 & C_x & 0 \\ 0 & \frac{f}{S_y} & C_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} {}^cT_w \begin{bmatrix} P^w \\ 1 \end{bmatrix} = M \begin{bmatrix} P^w \\ 1 \end{bmatrix} \tag{3-9}$$

M is a 3*4 parameter matrix, determined by the internal and external parameters of the camera.

In general the inversion of the camera model is not possible, the projection process transforms the 3D information to 2D and the distance information is lost.[44] As shown in Figure 3-6, the projection of each point on the image is the same, so the world coordinates cannot be determined from the image coordinates.

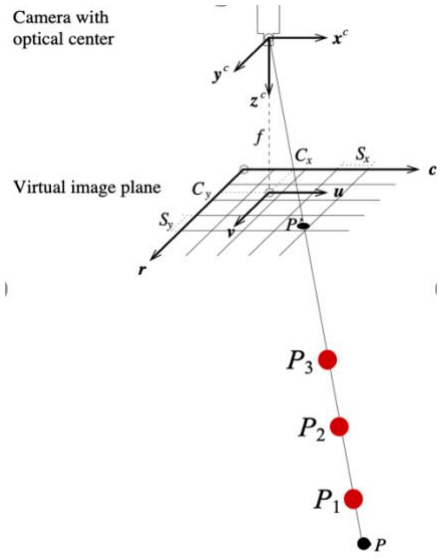


Figure 3-6 Inverse variation of the camera model

In this case, a camera with depth ranging function can be used to obtain distance information to complete the conversion from pixel coordinates to world coordinates.

ROS provides a number of tf (transform) related functions for coordinate changes. As long as the relationship between two coordinates is known, the user can directly call

the related functions to broadcast a custom coordinate change relationship, and other nodes can subscribe to such a tf relationship to complete the transformation of points in different coordinate systems. Other nodes can subscribe to such a tf relationship to transform points in different coordinate systems.

3.5 Convolutional neural network-based target detection

Deep learning target detection algorithms are classified into two categories: two-stage target detection algorithms and single-stage target detection algorithms according to the presence or absence of a candidate frame generation stage.[3] The algorithms are divided into two categories: two-stage target detection algorithms and single-stage target detection algorithms. Scholars usually measure the advantages and disadvantages of neural networks in terms of speed and accuracy, generally speaking, single-stage target detection neural networks have faster speed, while two-stage target detection networks have higher accuracy.[45] The current common single-stage target detection networks have higher speed and higher accuracy. Current common single-stage target detection networks include YOLO, SSD, etc., and two-stage neural networks include Faset-RCNN, Faster-RCNN, etc. Literature[46] It is pointed out in the literature that the SSD network uses 300*300 RGB images as training data, and the mAP of this model reaches 77.5% after training the model with COCO data set, which is better than YOLO and close to Faster R-CNN, so this paper uses the SSD model with both accuracy and speed as the detection model for target recognition. The comparison is shown in the following table.

Table 3-2 Comparison of different deep learning models mAP[46]

Methods	mAP
SSD	77.5
Faster R-CNN	75.9
YOLO	57.9
Fast R-CNN	68.4

Figure 3-7 shows the structure of the SSD network. The SSD method establishes the structure based on a feed-forward convolutional neural network, and adds numerous auxiliary structures after the VGG-16 base network structure to construct a feature generation perceptron

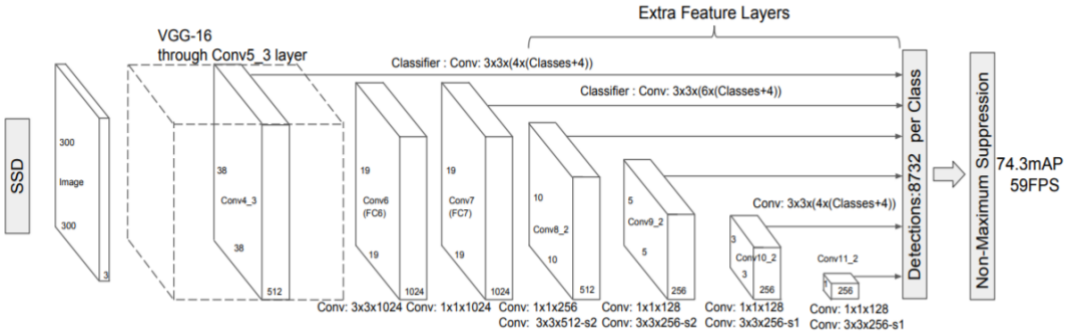


Figure 3-7 Structure of SSD deep learning neural network

Each multiscale feature block of the SSD network reduces the height and width of the feature map provided by the previous layer. These blocks then use each element of the feature map to expand the perceptual field on the input image. The closer a scale feature block is to the top, the smaller its output feature map is and the fewer anchor frames are generated based on the feature map. In addition, the closer a feature block is to the top, the larger the receptive field of each element in the feature map, and the more suitable it is for detecting larger objects. Since SSD generates different numbers of anchor frames of different sizes based on the underlying network blocks and each multi-scale feature block, the class and offset of the anchor frames are then predicted to detect objects of different sizes.

COCO is a large-scale object detection, segmentation and captioning dataset with 330,000 images, of which more than 200,000 have been annotated and can be used directly in supervised learning training. The dataset hosts 80 types of objects and has a wide range of applications.

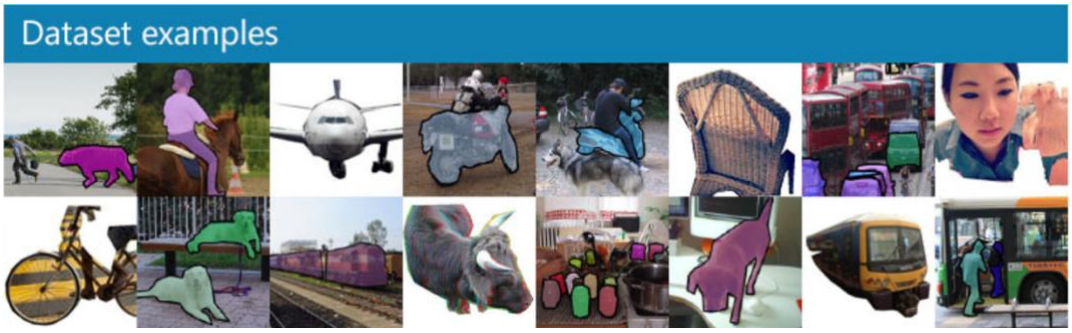


Figure 3-8 Example of CoCo dataset

After the training of the neural network is completed, the modules required for ROS are added, and the pixel coordinate points and depth values of the center of the object bounding box are extracted by subscribing to the color image information and depth

image information, and the 3D coordinate values of the object in the camera coordinate system are calculated using the distance formula, after which the ROS message (msg) format as shown in Table 3-3 is customized, and the object The ROS message (msg) format as shown in Table 3-3 is customized to publish the object type and positioning information as a topic for other nodes to use.

Table 3-3 Custom Message Format Pseudocode

Customized msg message format
1.string Types of objects
2. float32 x coordinate value
3. float32 y coordinate value
4. float32 z-coordinate value

3.6 Visual positioning error analysis

The effectiveness of the vision system is measured by placing CoCo data set training objects such as water glasses, cell phones, benches, computers, etc. in the camera's field of view.

Using the rqt tool that comes with the ROS system to visualize the vision system's real-time inspection screen, the image uses a bounding box to frame the object, and the depth value in the center of the bounding box is displayed visually on the image. The information output in the vision program terminal has information about the object type and its 3D coordinates. The number after the object type information represents the number of the object in the camera's field of view, this number starts from 0, where laptop stands for laptop and chair stands for chair, the overall overview diagram is shown below.

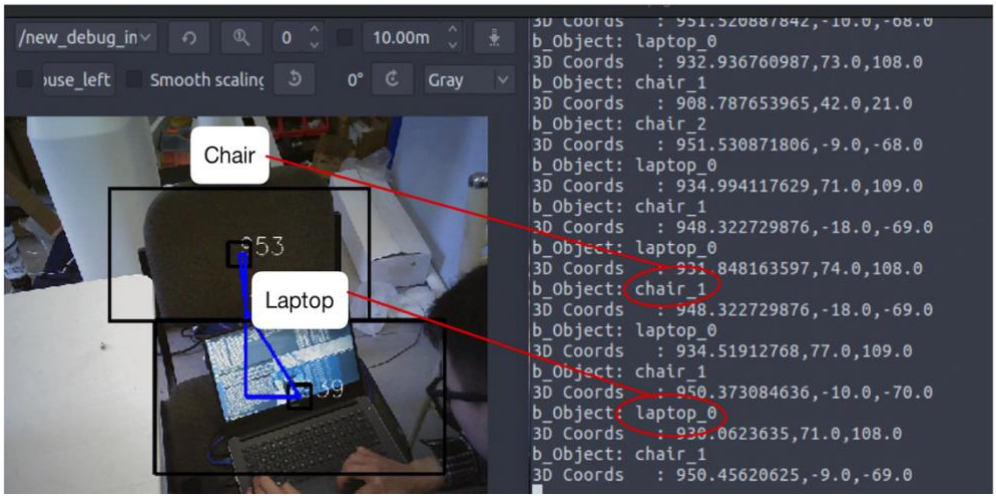


Figure 3-9 Visual recognition terminal

It is found that the accuracy of object location recognition is greatly affected by the light, and the work of the infrared sensor will be affected when the indoor light intensity is high resulting in the loss of depth information. After adjusting the indoor light to the appropriate range, the following terminal output and the actual measured depth comparison information were measured.

Table 3-4 Object coordinate information comparison table (unit:mm)

Object Name	Output	Actual Coordinates
Water Cup	(33.2, 31.7, 550)	(32, 31.5, 547)
Cell phone	(42.5, 41.6, 560)	(42, 41.5, 556)
Computer	(56.2, 49.7, 680)	(56.2, 49.7, 683)
Bench	(375, -42, 1086)	(364, -40, 1090)
Bottle	(426.1, -92, 997)	(420, -90, 1002)
Mouse	(327, 42, 860.5)	(325, 47, 860)
Keyboard	(497, 43, 762.1)	(495, 49, 760)

Table 3-4 in the output coordinates and the actual coordinates are calculated after five measurements, the analysis shows that the absolute error of positioning are within 1cm, the reason for the error may be the influence of the environment, such as the light source, the color of the object, etc., may also be caused by the distance measuring principle of the camera itself, because this paper only study the dual-arm robot to grasp larger objects such as water cups, bottles, so the error can be considered within the acceptable range. The error can be considered to be within the acceptable range.

Due to the thickness of the object itself and the fact that the vision system uses the method of extracting depth information from the center of the bounding box to calculate

the 3D coordinate system leads to a centimeter offset between the localization point and the optimal grasping point, which can be reduced later through experimental testing.

To obtain the object centroids more accurately one can use the method proposed in the literature[24] The proposed method uses ORB descriptors to confirm the target contour and match the centroids for the problem that the detection accuracy of neural networks may be low and the object contour centroids are not estimated accurately.

In order to exclude abnormal localization values and reduce the impact of errors on the final experimental results, the vision system records the coordinate information of the same object collected five times as shown below, excludes the values that are not within the mean plus or minus double the standard deviation, and uses the mean value of the last unexcluded point as the final published 3D coordinate point. The method is codified using the numpy library in the python library.

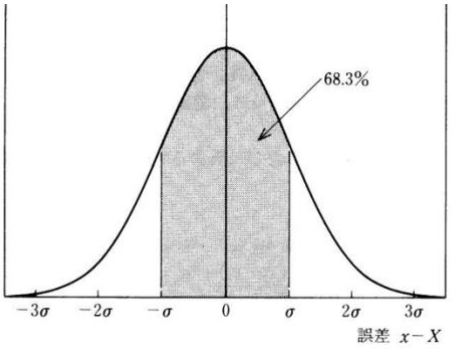


Figure 3-10 Confidence interval

3.7 Summary of this chapter

This chapter introduces the selection of camera, the principle of internal parameter calibration and its implementation, the selection of deep learning model for object recognition and its implementation, and finally the experimental test of the whole vision system, which analyzes the causes of the localization error and proposes a method to reduce the localization error applicable to this vision system.

4 Dual-arm collaboration and intelligent control

This part adopts ROS (Robot operating system) to realize the two-arm collaboration and intelligent control of the robot, which depends on the same environment as the vision system.

4.1 Creating the robot model file

Create the urdf file based on the parameter information of the real robot, and make the joint coordinate system of the robot in the simulation environment consistent with the joint coordinate system of the real robot. In Figure 4-1, xyz represents the translation relationship of each coordinate system, rpy represents the rotation relationship, base_link is the robot base coordinate system, and the prefix r/l refers to the joints and links of the left and right arms, respectively.

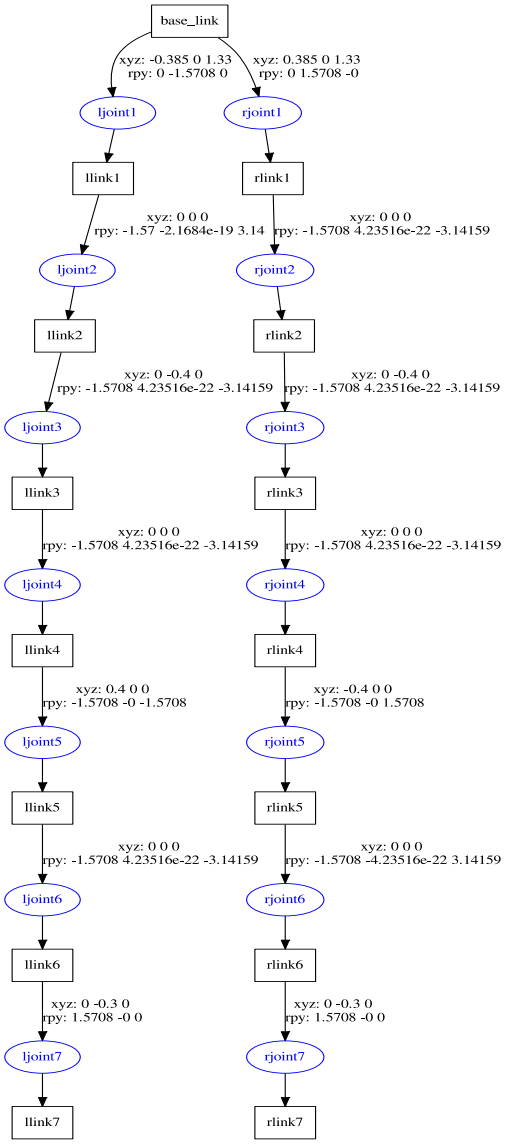


Figure 4-1 Robot URDF file parameter structure

The Rviz visual debugging interface is used to observe the trajectory and joint positions of the robot in the simulation environment, compare the execution of the real robot, and make the two consistent through debugging. The urdf model of the robot can be checked for correctness with the ROS visualization plug-in.

Use MoveIt Setup Assistant to configure the robot, add collision parameters to the robot in the virtual environment, add the left_manipulator and right_manipulator groups, preset the initial pose of the robot, preset the inverse solver to IK_FAST, add the ROS control motion controller, and generate a series of .launch/.yaml/CmakeList/package initial files.

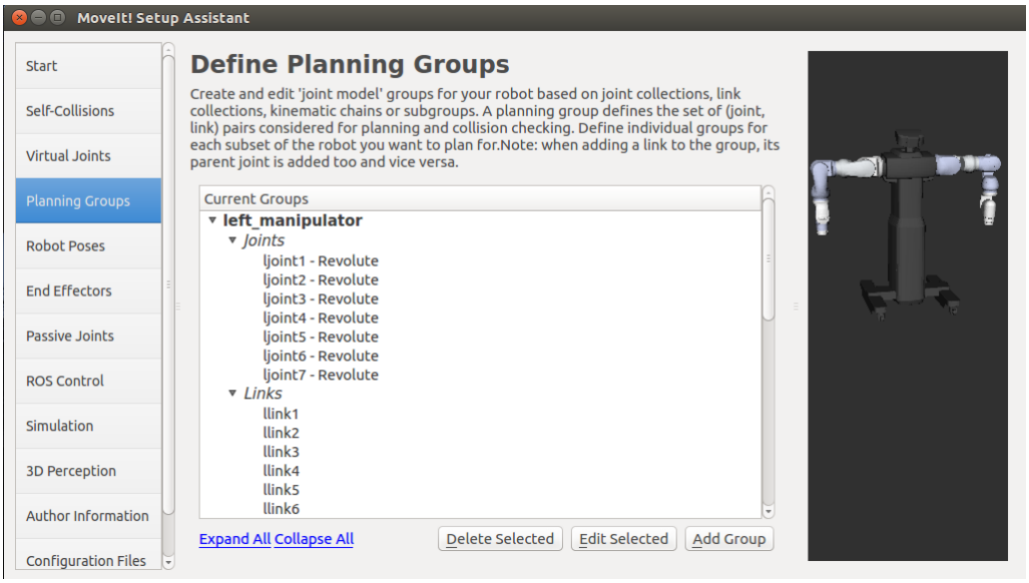


Figure 4-2 MoveIt visual configuration window

4.2 Hand-eye calibration principle and its implementation

In the previous chapter, the internal parameters of the vision system have been calibrated to reduce the effects of aberrations, in order to obtain the relationship between the camera coordinate system and the robot base coordinate system, hand-eye calibration is also required[47] .

According to the location of the camera can be divided into two categories of robot hand-eye system, one is Eye-in-Hand, this kind of system is mounted on the robot arm, the other is Eye-to-hand, this kind of system is mounted on the top of the robot arm[27] . This article uses the Eye-to-hand hand-eye system, and the calibration of this system will be introduced next.

Assuming that the camera coordinate system is C and the base coordinate system of the machine is W , assuming that the end joint coordinates and the gripper coordinates are the same, after fixing the calibration plate on the robot arm gripper, using the calibration plate coordinates instead of the robot arm end-effector coordinates, the position of the calibration plate coordinate system relative to the camera coordinate system and the position of the robot arm end-effector relative to the camera coordinate system have the same transformation relationship. The relationship between the robotic arm end-effector and the robot base coordinate system can be derived from the kinematic derivation of the robot so that the calibration plate moves within the visual range of the

camera, the coordinates of the corresponding point on the calibration plate under the base coordinate system and the coordinates of the calibration plate under the camera coordinate system are recorded, and the external parameter matrix of the camera is derived by the hand-eye calibration algorithm [21] The Eye-to-Hand system is shown in Figure 4-3.

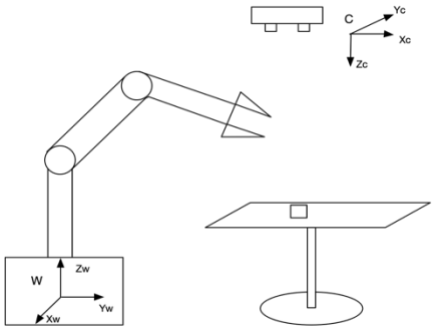


Figure 4-3 Eye-to-hand diagram

Assuming that$(X^c \ \ Y^c \ \ Z^c)^T$ is the point under the camera coordinate system of the marker point in calibration plate z and$(X^w \ \ Y^w \ \ Z^w)^T$ is the point of the marker point under the coordinate system of the robot arm, their correspondence can be expressed by the equation

$$\begin{bmatrix} X^c \\ Y^c \\ Z^c \\ 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0\,0\,0 & 1 \end{bmatrix} \begin{bmatrix} X^w \\ Y^w \\ Z^w \\ 1 \end{bmatrix} = \begin{bmatrix} r_1 & r_2 & r_3 & t_x \\ r_4 & r_5 & r_6 & t_y \\ r_7 & r_8 & r_9 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X^w \\ Y^w \\ Z^w \\ 1 \end{bmatrix} \tag{4-1}$$

After n times of calibration Eq.(4-1) can be expressed as

$$\begin{bmatrix} X_1^w & Y_1^w & Z_1^w & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & X_1^w & Y_1^w & Z_1^w & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & X_1^w & Y_1^w & Z_1^w & 1 \\ X_2^w & Y_2^w & Z_2^w & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & X_2^w & Y_2^w & Z_2^w & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & X_2^w & Y_2^w & Z_2^w & 1 \\ & & & & & & \cdots & & & & & \\ X_n^w & Y_n^w & Z_n^w & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & X_n^w & Y_n^w & Z_n^w & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & X_n^w & Y_n^w & Z_n^w & 1 \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ t_x \\ r_4 \\ r_5 \\ r_6 \\ t_y \\ r_7 \\ r_8 \\ r_9 \\ t_z \end{bmatrix} = \begin{bmatrix} X_1^c \\ Y_1^c \\ Z_1^c \\ X_2^c \\ Y_2^c \\ Z_2^c \\ \vdots \\ X_n^c \\ Y_n^c \\ Z_n^c \end{bmatrix} \tag{4-2}$$

The above equation can be decomposed into the following equations.

$$\begin{bmatrix} X_1^w & Y_1^w & Z_1^w & 1 \\ X_2^w & Y_2^w & Z_2^w & 1 \\ & \cdots & & \\ X_n^w & Y_n^w & Z_n^w & 1 \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ t_x \end{bmatrix} = \begin{bmatrix} X_1^c \\ X_2^c \\ \cdots \\ X_n^c \end{bmatrix} \tag{4-3}$$

$$\begin{bmatrix} {X_1}^w & {Y_1}^w & {Z_1}^w & 1 \\ {X_2}^w & {Y_2}^w & {Z_2}^w & 1 \\ & \cdots & & \\ {X_n}^w & {Y_n}^w & {Z_n}^w & 1 \end{bmatrix} \begin{bmatrix} r_4 \\ r_5 \\ r_6 \\ t_y \end{bmatrix} = \begin{bmatrix} {Y_1}^c \\ {Y_2}^c \\ \cdots \\ {Y_n}^c \end{bmatrix} \tag{4-4}$$

$$\begin{bmatrix} {X_1}^w & {Y_1}^w & {Z_1}^w & 1 \\ {X_2}^w & {Y_2}^w & {Z_2}^w & 1 \\ & \cdots & & \\ {X_n}^w & {Y_n}^w & {Z_n}^w & 1 \end{bmatrix} \begin{bmatrix} r_7 \\ r_8 \\ r_9 \\ t_z \end{bmatrix} = \begin{bmatrix} {Z_1}^c \\ {Z_2}^c \\ \cdots \\ {Z_n}^c \end{bmatrix} \tag{4-5}$$

By observation it can be seen that all the above equations are $Ax = b$, where A matrix dimension is $n \times 4$, x is 4×1 , b is $n \times 1$, and the matrix A can be decomposed into a regular orthogonal matrix Q and an upper triangular matrix R , i.e.

$$A = QR \tag{4-6}$$

Then the solution of the linear equation $Ax = b$ is

$$X = R^{-1}Q^{-1}b \tag{4-7}$$

Separately to the equation(4-3) ,(4-4) , and(4-5) Performing the QR decomposition, i.e., the transfer matrix can be solved as

$$H = \begin{bmatrix} r_1 & r_2 & r_3 & t_x \\ r_4 & r_5 & r_6 & t_y \\ r_7 & r_8 & r_9 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4-8}$$

In the hand-eye system required in this paper, the combined form of Eye-to-hand can make the camera get a better field of view, and the stability of the image information can be guaranteed[48] Therefore, the Eye-to-hand system is used in this paper. The hand-eye calibration can be implemented through the open source code Easy_handeye on Github, which samples the robot position and tracking system output through tf. Through the OpenCV library's Tsai-Lenz[49] algorithm is implemented to compute the calibration matrix of the eye-to-robot base coordinate system or eye-to-hand and to store the results of the calibration. At each subsequent start-up, the results of the calibration procedure are published as tf transformations in the ROS system for subsequent use.

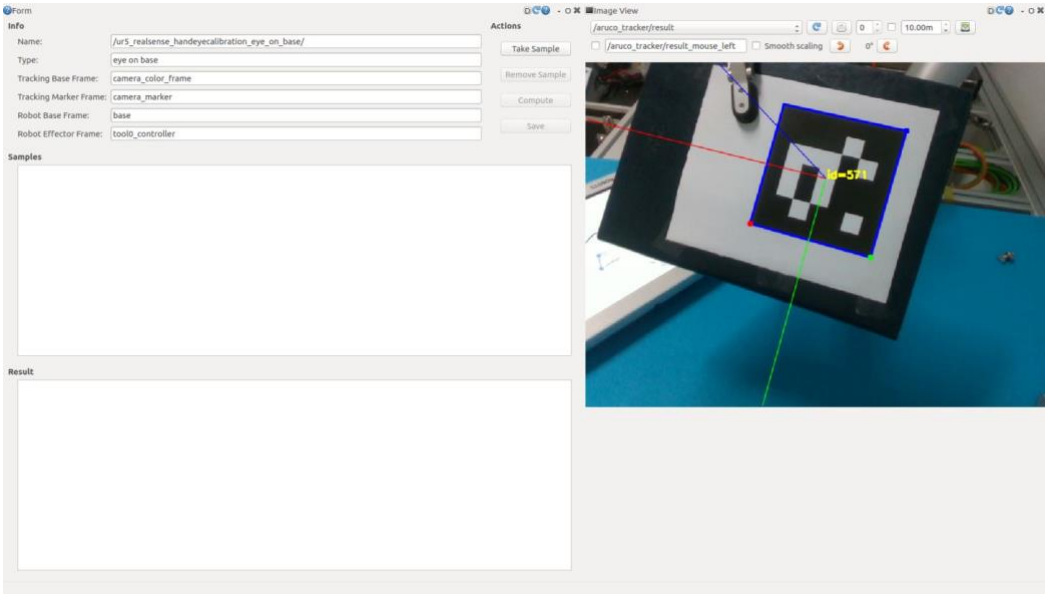


Figure 4-4 Hand-eye calibration program interface

The program subscribes to the coordinate change relationship (tf) from the robot's base scale to the end-effector and the image topic. After printing the calibration plate, the arm with the plate is moved to a different position and repeated tens of times to obtain a .yaml file with external parameters.

After the parameters are included in the robot's launch file and the hand-eye calibration, the camera coordinate system is displayed visually in the Rviz simulation environment, with red, green and blue representing the positive direction of the xyz axis respectively. The relationship between the camera coordinate system, the robot base coordinate system, and the joint coordinate system can also be clearly observed through this interface.

After the relationship between the camera coordinate system and the robot base system is determined, the coordinate system of the two-arm robot and its vision system as a whole is established. 18 coordinate systems are shown in Figure 4-5, including 14 coordinate systems of the two-arm joints, 2 gripper coordinate systems coinciding with the end joint coordinate system, 1 camera coordinate system, and 1 base system of the two-arm robot. The coordinate system of the robot in the simulation environment is consistent with that of the real robot, so that the camera coordinate system is located at a horizontal distance of 1.2 m and a vertical distance of 1.25 m from the base scale system of the robot, and the X-axis is kept at an angle of 45 degrees from the horizontal plane.

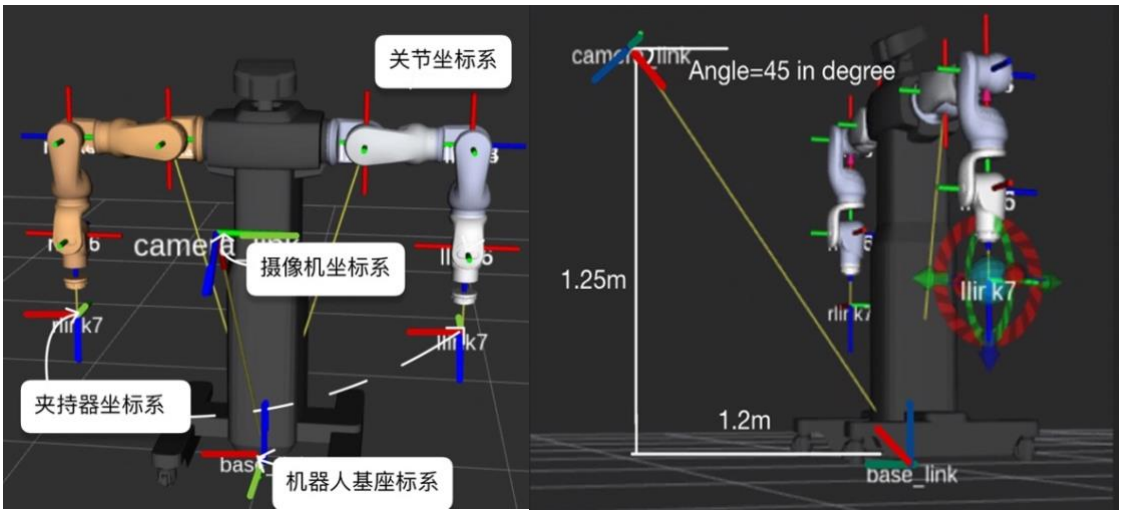


Figure 4-5 Schematic diagram of the overall coordinate system of the hand-eye system

4.3 Task-based collaborative motion planning with two arms

4.3.1 Virtual scene construction and motion planning

In order to make the two-armed robot avoid obstacles when performing tasks, this paper uses the built-in function of MoveIt to add collision objects, creates a rectangular body with a length and width of 80cm and a thickness of 3cm, this object is an abstraction of a desktop in a real environment, and restricts the planning space of the robot to a space of 2m*2m*2m. By subscribing to the 3D coordinate information and kind information of the object and the tf relationship obtained from the topic generated after hand-eye calibration, the 3D coordinate points of the object under the camera coordinate system are transformed into 3D coordinate points under the base coordinate system, and the task contents of the left and right arms are distinguished by the positive and negative x-values under the robot coordinate system, and the coordinate points of the objects located on both sides of the robot are designated as the planning target endpoints of the robot gripper, respectively.

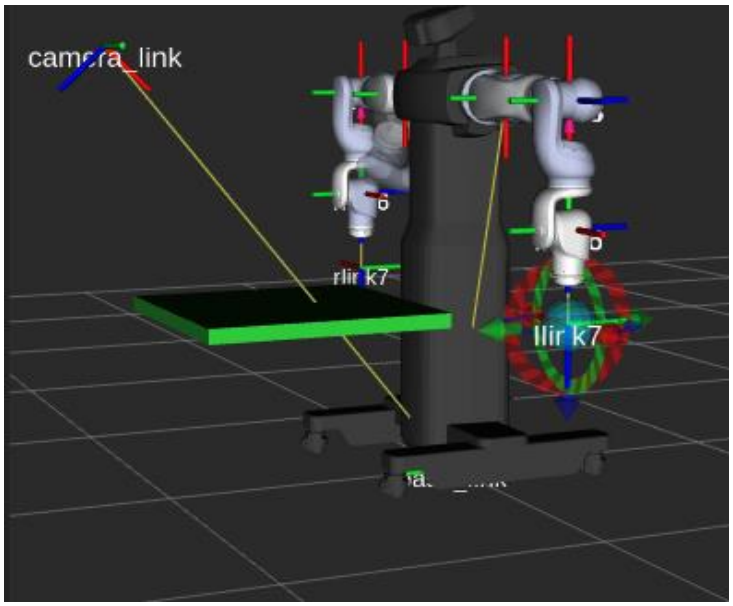


Figure 4-6 Virtual environment scene setting

The initial state of the real robot and the robot in the simulation environment are kept the same, the motion path of the robot from the initial position to the position to be gripped is calculated by TRAK-IK integrated in MoveIt, the pose of the robot gripper to grasp the object is preset, and since the action of pouring water after gripping the object is fixed, the script program movej_pose is used directly to develop the end position and Since the action of pouring water after gripping the object is fixed, the end position and pose are programmed directly using the script movej_pose for the action after gripping the cup and for the recovery.

4.3.2 Kinematic solver

To calculate the joint rotation information, a kinematic solver is needed, and ROS comes with several kinematic solvers, such as KDL, TRAK-IK, IKFAST, etc., which the user can choose according to his needs. The final result is a very stable solution that can be computed in the latest processors with a speed of 5 microseconds.

The joint-constrained pseudo-inverse Jacobian implementation of KDL is used as a solver for generic manipulation chains by various ROS packages and MoveIt! Researchers at TRACLabs encountered a large number of solution errors when using the inverse kinematic functions of KDL on a robotic arm. They attributed these problems to the fact

that theoretically sound Newtonian methods fail in the face of joint constraints. As a result, the developers created TRAC-IK to run two different IK methods simultaneously.

TRAK-IK is an enhanced version of the KDL solver that detects and mitigates the local minima that may occur when the joint limit is encountered during gradient descent. The formulation uses a quasi-Newton method that can better handle non-smooth search spaces. By combining these two methods, TRAC-IK outperforms the two independent IK methods with no additional runtime overhead for small chains and a significant time improvement for large chains.

As shown in Figure 4-7, the red part of the KDL solver shows the locations where the solution cannot be found, and the white part shows the parts where the solution can be found, so the comparison shows that TRAC-IK can find the kinematic solution in more locations.

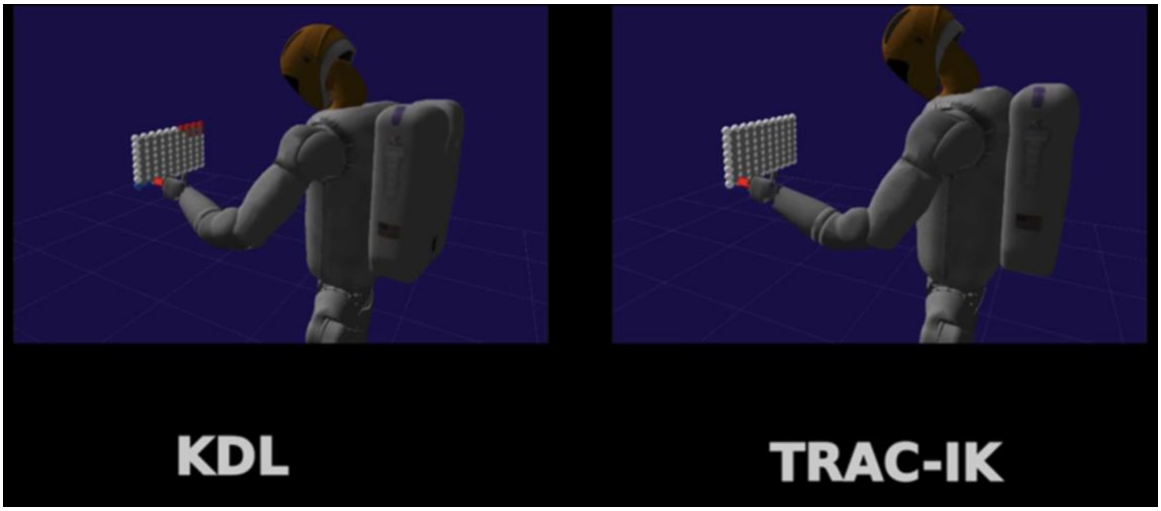


Figure 4-7 Comparison of KDL & TRAC-IK solution results

In this paper, we use IK-FAST to solve the kinematics of the robot. The algorithm of IKFAST uses openRAVE to generate the kinematics plug-in for cpp. The two-armed robot requires splitting the robot's model file urdf file into two urdf files for the left and right arms and converting the urdf file into a dae format file using the MoveIt plug-in. After setting the degrees of freedom of the robot and the name of the exported file, the program will automatically solve the kinematic equations of the robot and export the kinematic plug-in. Modify the path of the plug-in in the robot kinematics configuration file to IKFAST to complete the configuration of the kinematics plug-in.

4.4 Closed-loop control of two-arm collaboration based on vision perception

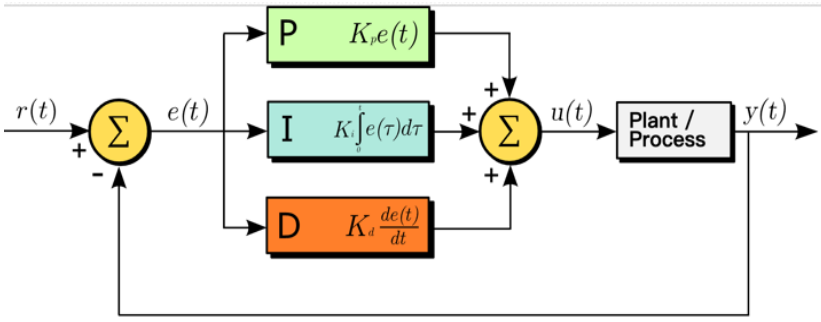
4.4.1 PID control

PID control is a control loop mechanism that uses feedback and is widely used in industrial control systems and various other applications that require continuous modulated control. the PID controller continuously calculates the error value as the difference between the desired set point (SP) and the measured process variable (PV) and corrects for the proportional, integral and derivative terms (denoted P, I and D, respectively).[50] .

P is proportional to the current value of the SP-PV error e(t). For example, if the error is large and positive, the control output will be proportionally large and positive considering the gain factor "K". Using only proportional control results in an error between the set value and the actual process value because it requires an error to produce a proportional response. If there is no error, there is no corrective response.

I takes into account the past SP-PV error values and integrates them over time to produce the term I. For example, if there is a residual SP-PV error after the application of proportional control, the integration term eliminates the residual error by adding a control effect due to the accumulated historical value of the error. When the error is removed, the integral term will stop growing. This will cause the proportional effect to decrease as the error decreases, but this will be compensated by the growing integral effect.

D is an estimate of the future trend of the SP-PV error, based on its current rate of change. It is sometimes referred to as "anticipatory control" because it effectively seeks to reduce the impact of SP-PV error by imposing a control influence generated by the rate of change of the error. The faster the change, the greater the control or damping effect.



4-8 PID control schematic

ROS has an integrated PID control package, this package runs as a node in ROS and the PID controller subscribes and publishes the following topic names. The setpoint, state, and control_effort topic names are default and can be changed in a standard way through ROS techniques such as remapping and namespaces, or in a pid package specific way through controller parameters.

(1) setpoint, the controller subscribes to the topic and reads the std_msgs/Float64 message. The message data element contains the expected value of the state measurement of the controlled process.

(2) state, the controller subscribes to this topic and reads the std_msgs/float64 message. The message data element contains the current value of the controlled robot joint property. The controller publishes std_msgs/float64 messages on the control_effort topic each time it receives a message from the state topic.

(3) control_effort: The control_effort message data element contains the control applied to the process, driving the state/data to equal the setpoint/data.

(4) pid_enable: The controller subscribes to this topic and reads std_msgs/Bool messages. If this value is false, the controller stops publishing control_effort and keeps the error credit at 0. A real value re-enables the controller. If you have multiple PID controllers, the name of this topic can be changed in the startup file.

(5) pid_debug: Publish an array that is useful for debugging or tuning. The array contains five numbers: error, control effort, proportional contribution, integral contribution, and derived contribution.

Closed-loop feedback control of the robot is performed using the system preset parameters. The real joint information of the robot is sent from the 2001 port under the robot ip, which is encapsulated as /state information for the PID system to subscribe. Making the value of pid_enable True, the closed-loop control block diagram of this system is shown in Figure 4-9.

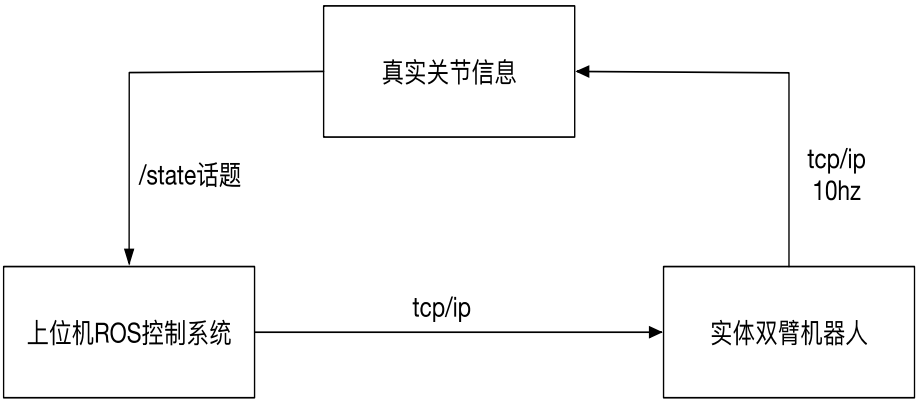


Figure 4-9 Robot system control block diagram

In this paper, there are two basic controllers that are affected by the PID parameters: the joint velocity controller and the joint position controller. Taking the joint position controller as an example, the comparison graph before and after the PID parameter adjustment is shown in Figure 4-10. Before adjusting the PID controller, the robot arm cannot reach the set pose.

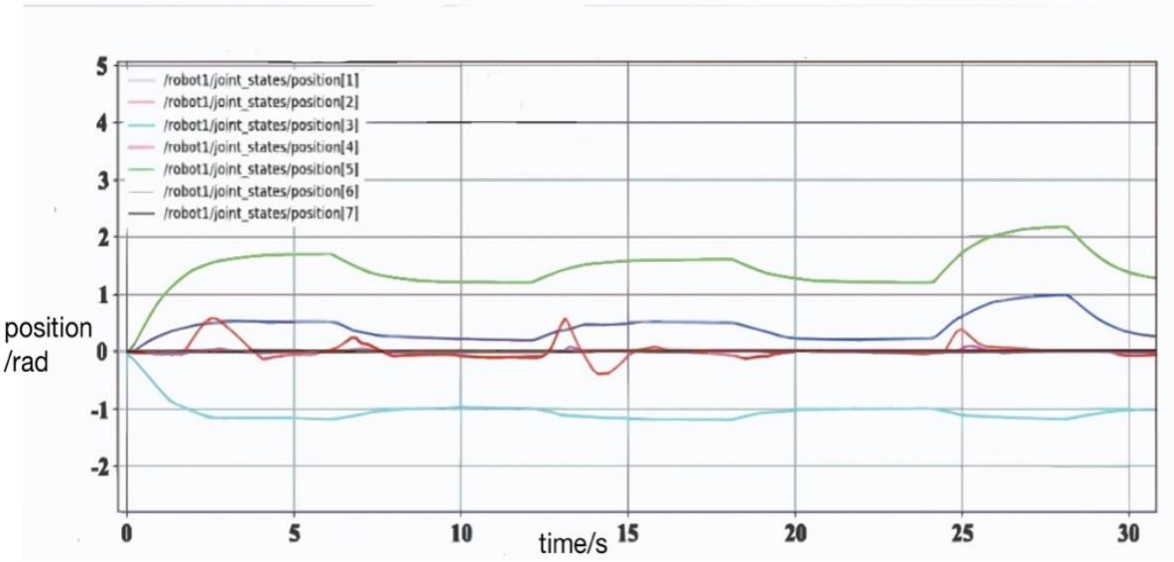


Figure 4-10 Joint position before PID modulation

Before adjusting the PID controller, the rotation angle of joint 2 often exceeded the given range, and its maximum value even exceeded the rotation angle of joint 1. The joint positions after adjusting the PID controller are shown in Figure 4-11, and the performance of the robot arm is significantly improved. The adjusted controller can meet the requirements of this paper and make the robot arm achieve the set pose during the simulation.

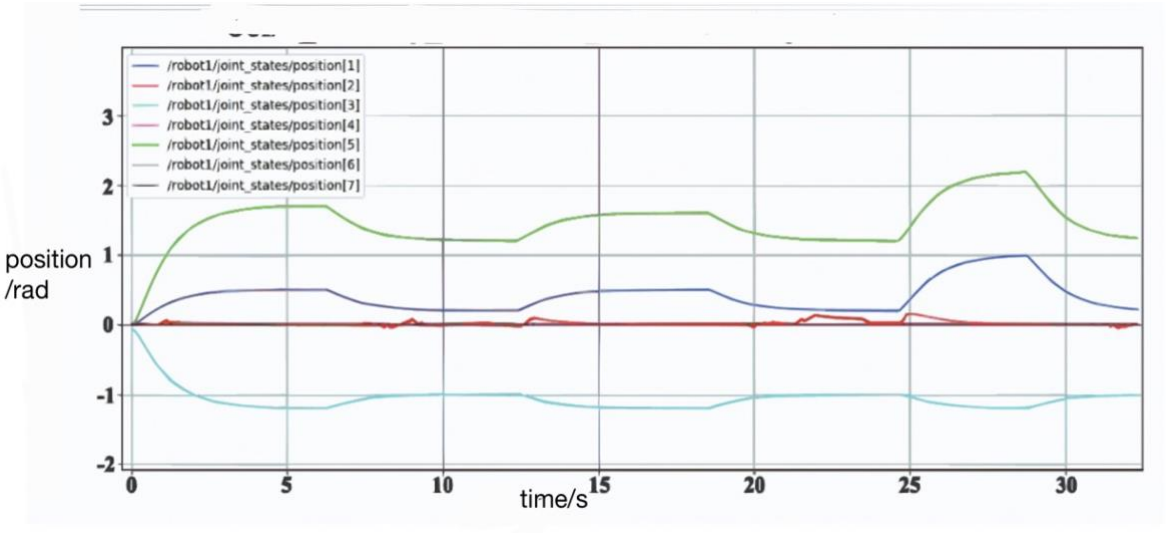


Figure 4-11 Joint position after PID modulation

4.4.2 ROS output information utilization

The action communication format is characterized by a feedback section, which provides feedback from the robot on joint information, joint force information, task execution messages, and so on. After sending a task goal to the physical robot, the robot will provide real-time feedback on the task execution, and the user can cancel the task during execution and return the completion flag when the task is completed.

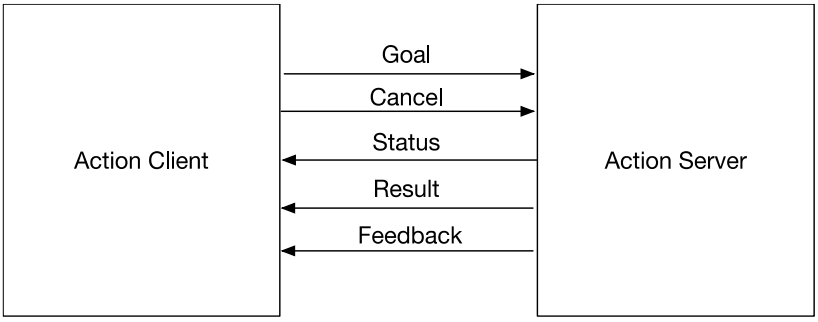


Figure 4-12 action communication principle

The controller used by the robot in the MoveIt scenario is the server side of an action called Follow Joint Trajactory. The user needs to write the client side of the action to receive the output information, which contains the timestamp, acceleration, velocity, and rotation angle of each joint.

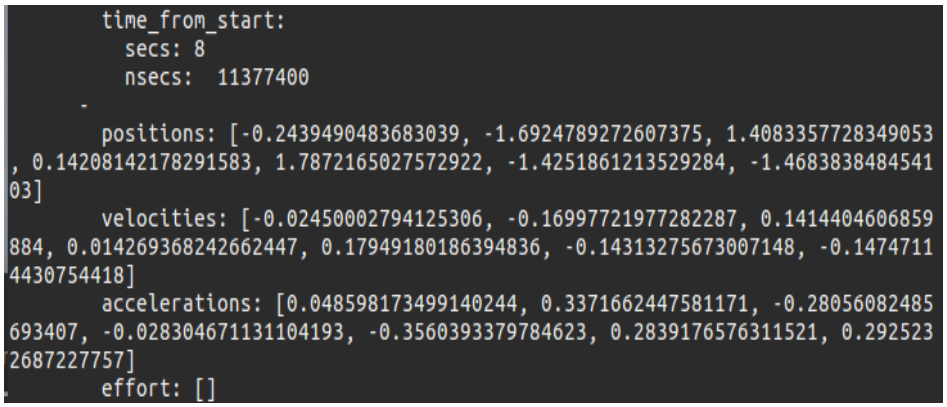


Figure 4-13 ROS system output information

By extracting the position information of the joints and using C++ string splicing functions to encapsulate the planning information, the robot will encapsulate the information into the format of script functions that can be recognized by the robot's own system, such as movej (move the joint), move_pose (move to the specified position), etc. When the robot finishes executing the script functions, it will send a 'script finish' message to the host computer, when the host computer receives this signal, it will send 'excuteOpen' or 'executeClose' string to the Raspberry Pi. Raspberry Pi is in the training state, the program will enter the 'if' statement to execute the corresponding action of opening or closing the gripper, before executing the opening or closing action, it will display the message of about to open or about to close on the terminal.

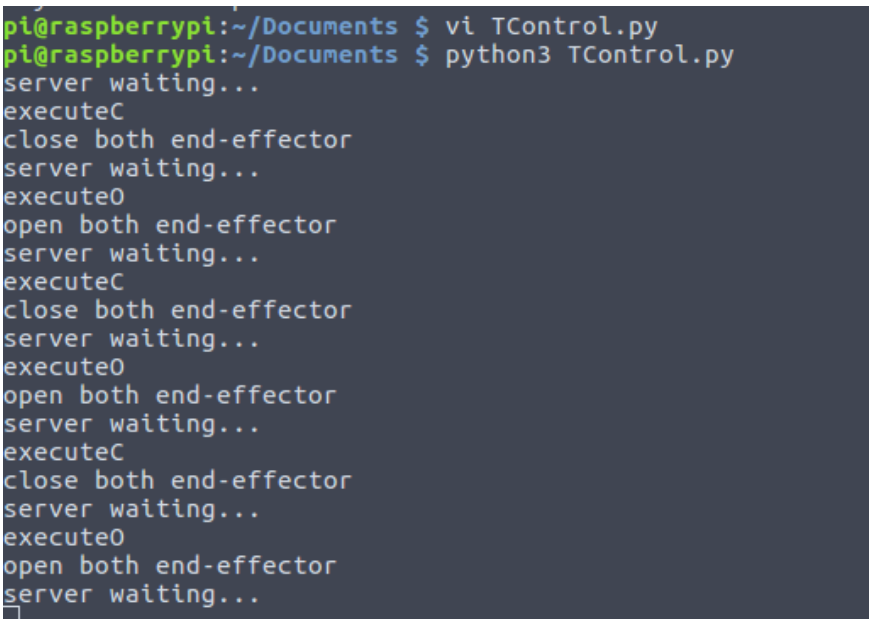


Figure 4-14 Raspberry Pi terminal prompt message

The robot script functions start with <start> as the program start flag, /LP and /RP are used to distinguish between left and right arm programs, each script function ends with ';', and the script program ends with <end> as the program end flag. The main robot scripting functions used in this system are movej, movej_pose, sync_flag, and sleep functions, which are described next.

（1） movej, which moves to the specified joint position by means of joint angle motion, requires the user to give the rotation angle of each joint.

（2） movej_pose, specifies the position and pose of the end-effector, moving to the specified position pose.

（3） sync_flag, the program will continue to execute when both the left and right arm scripts move to the same sync_flag.

（4） sleep, the delay function.

The robot supports the following data types.

(1) Arithmetic type: The system built-in arithmetic type includes integer (int) and floating point (float) data, which are represented in the system internal 32bit memory space. The system automatically infers the type of arithmetic type constants (based on whether the constants contain decimal points or not), and the system variables are defined at the time of definition to determine the type based on the initialization content.

(2) String type string: The system only supports constants of string type, such as "siasun robot", and the string ends with '\0'.

(3) Pose: A pose is a float vector of length 6. The system supports constants and variables of the pose type.

(4) Array type: The system supports two types of built-in array types, int and float.

Users can also program the robot using common while and break functions, assignment expressions, function call expressions, etc.

```
<start>LP/
movej([-0,-0,-0,-90,0,-0,0],30,2000,-1); movej([5,5,-7,-87,11,7,13],30,2000,-1);
movej([10,11,-14,-84,22,14,26],30,2000,-1); movej([15,16,-22,-81,34,21,39],30,2000,-1);
movej([20,21,-29,-78,45,27,53],30,2000,-1); movej([25,27,-36,-76,56,34,66],30,2000,-1);
movej([30,32,-43,-73,67,41,79],30,2000,-1); movej([35,37,-50,-70,79,48,92],30,2000,-1);
movej([40,43,-58,-67,90,55,105],30,2000,-1); movej([45,48,-65,-64,101,62,118],30,2000,-1);
movej([50,53,-72,-61,112,69,132],30,2000,-1); movej([55,59,-79,-58,124,75,145],30,2000,-1);
movej([61,64,-86,-55,135,82,158],30,2000,-1); movej([66,69,-94,-53,146,89,171],30,2000,-1);
movej([67,71,-95,-52,148,91,174],30,2000,-1);movej_pose([-497,584,1112,78,51,90],40,20,-1);
movej_pose([-327,584,872,78,51,90],40,20,-1);
/RP/ movej([0,-0,0,90,-0,-0,0],30,2000,-1);
movej([10,-1,1,83,13,-8,13],30,2000,-1); movej([19,-2,2,77,25,-15,27],30,2000,-1);
movej([29,-3,3,70,38,-23,40],30,2000,-1); movej([38,-4,4,64,51,-30,54],30,2000,-1);
movej([48,-5,5,57,63,-38,67],30,2000,-1); movej([58,-6,6,51,76,-45,81],30,2000,-1);
movej([67,-7,7,44,88,-53,94],30,2000,-1); movej([74,-1,9,40,98,-62,107],30,2000,-1);
movej([75,14,13,39,104,-73,120],30,2000,-1); movej([76,30,17,38,110,-84,133],30,2000,-1);
movej([77,46,22,37,115,-95,146],30,2000,-1); movej([78,58,25,36,120,-103,157],30,2000,-1);
movej_pose([525,648,1113,90,40,-95],40,20,-1);movej_pose([355,648,873,90,40,-95],40,20,-1);
<end>
```

Figure 4-15 Example of a script program

The real robot will act as a socket server after powering up to accept the encoded string commands sent to it by the user through the socket client, and the robot will also distribute real joint status information through port 2001 at a frequency of 10hz. By writing the Socket client program, this program is also the action client program used to receive waypoints from the virtual environment path planning, which can achieve the function of sending waypoints to the real robot. The real robot will automatically create a .spf file to store and immediately execute this script program after receiving the script function format string.

After connecting to port 2001 under the ip address of the built-in system of the robot, the real joint information can be read and fed back to the simulation system to update the position of the robot arm in the simulation system. Comparing the rotation angle information of the real joints in the demonstrator with the joint information of the last waypoint in the sent script function, the angles of a total of 14 joints of the two arms are exactly the same. The motion trajectory of the real robot is also exactly the same as the motion plan of the simulated robot.

4.4.3 Block diagram of the robot control system

The system hardware composition of the robot in this paper includes a depth camera, a host computer, a Raspberry Pi, a gripper, and a two-arm robot body. The depth camera is responsible for the image information acquisition, the host computer controls the movement of the robot arm, the Raspberry Pi controls the opening and closing of the gripper, and the dual-arm robot and the gripper are responsible for the execution of the

movement.

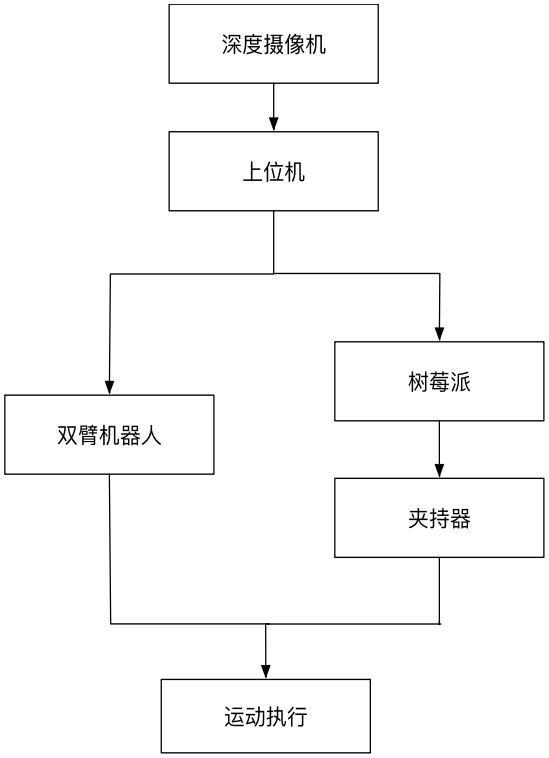


Figure 4-16 Hardware system block diagram

The software control flow of this paper is that the color image information and depth image information captured by the depth camera are processed by Tensorflow to derive object category information and 3D coordinate information exported as "/LeftObject" and "/ RightObject" topics, the path planning program subscribes to the above two topics as the information input for the upper computer control system MoveIt path planning, adds obstacles to the planning scene, the robot model file uses the exported SRDF parameter file for path planning and other necessary information, the solver planning process and outputs the waypoint results to the two-arm The solver outputs the waypoint results to the two-armed robot using python string processing functions to encapsulate the waypoint information into the robot scripting language. The real joint information is then encapsulated into the ROS topic '/state', and the PID control node of the host control system is subscribed to the '/state' topic to form closed-loop control.

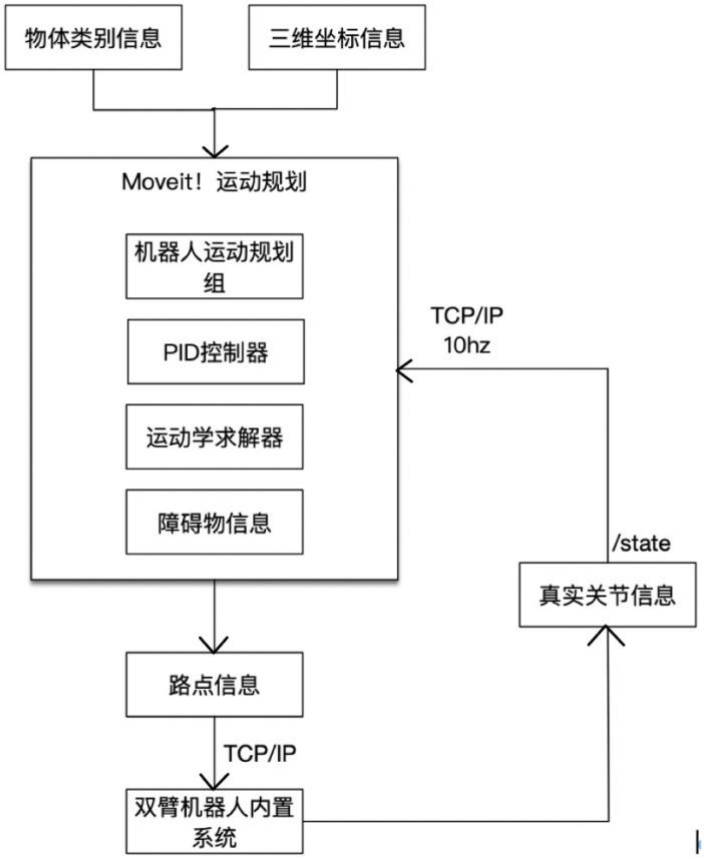


Figure 4-17 Software system block diagram

4.5 Summary of this chapter

This chapter simulates the motion of a real robot and its operating environment by building a robot model urdf file, adding collision objects using MoveIt's function interface, and limiting the planning space. The PID controller is added to the ROS system, the topic of subscribing visual information to integrate the hand-eye system, and the joint rotation data information output from the ROS is transformed into the format of script functions recognizable by the robot through python/C++ string processing related functions, and the processed strings are sent to the robot through socket network communication to realize the movement of the physical robot.

5 Experimental validation of a two-armed collaborative robot

In this chapter, three task scenarios will be set: single-arm motion, clamping and gripping objects, and collaborative gripping of objects by both arms. A depth camera will be used to capture image information for input to the host system, and a ROS-controlled solid robot will be used to test the effectiveness of the two-arm robot hand-eye system.

5.1 Single-arm motion experiment

The output information of ROS converts the waypoint information into the angle value of each joint, and uses the upper computer to send joint rotation information to the single arm of the robot via Socket. Since the simulation outputs large joint information, and the robot used in this paper can accept up to 8192 bytes of string in a single pass, in order to limit the size of the sending script function, the joint values are sampled at intervals and then added to the last To limit the size of the sending script function, the joint values are sampled at intervals, and then the joint values of the last target point are added to form a new set of joint values. Comparing the joint angle values sent to the robot (Figure 5-1) with the real joint angle values fed back by the robot (Figure 5-2), the two trends are basically the same.

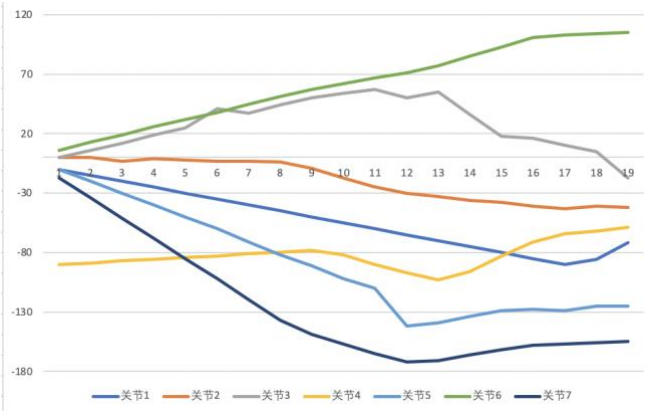


Figure 5-1 Sent joint rotation angle curve graph

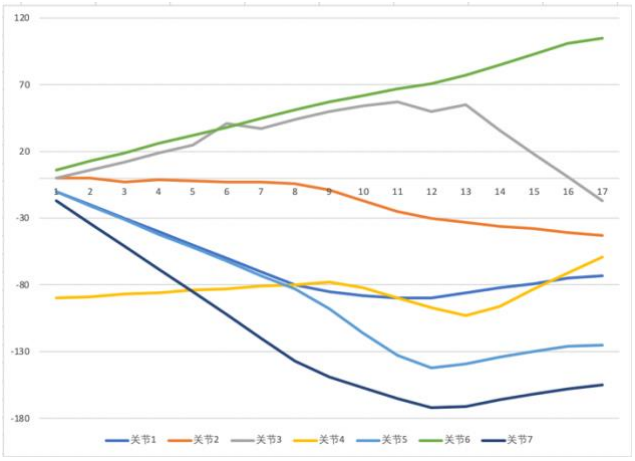


Figure 5-2 Real joint rotation angle curve

In order to more visually compare the motion trajectory of the robot in the simulated and real environment, we use rviz to visualize the motion trajectory of the single arm of the two-armed robot in the simulated environment and compare it with the motion of the physical robot.

Figure 5-3 shows the motion trajectory in the rviz simulation environment.

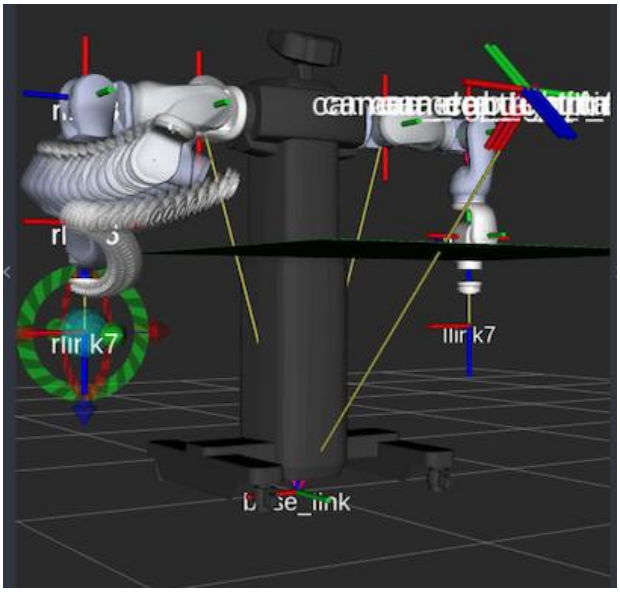


Figure 5-3 Simulation environment motion planning trajectory

Figure 5-4 shows the motion trajectory of the solid robot.

Figure 5-4 Motion trajectory of the solid robot

The comparison analysis shows that the motion trajectory of both robots is basically the same. In the actual test, the motion of the robot in the simulated environment is more smooth, while the motion of the solid robot has a millisecond pause after reaching a certain way point, and then moves to the next way point. It is assumed that the reason for this pause is the response speed of the robot execution program.

5.2 Grabbing experiments

After the gripper is assembled and stabilized by the connector and the robot, the upper computer and the Raspberry Pi that controls the movement of the gripper are put under the same WIFI network, and the stepper controller of the gripper and the GPIO of the Raspberry Pi are connected using a DuPont cable.

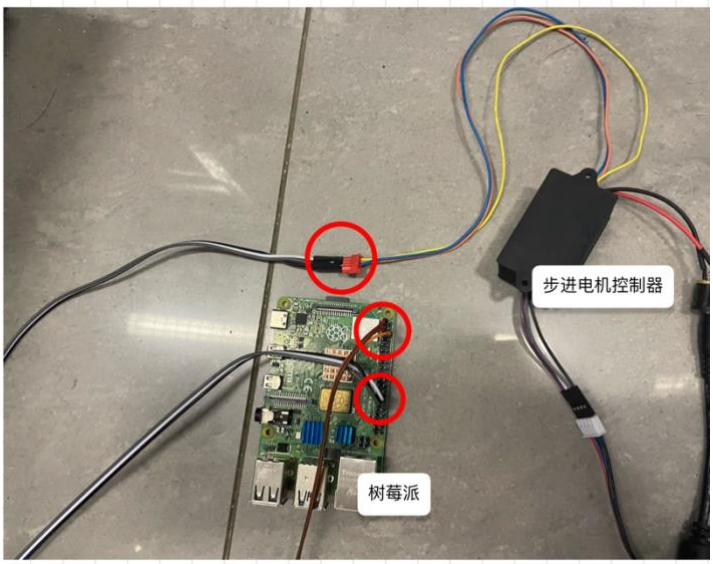


Figure 5-5 Raspberry Pi Connections

The stepper motor controller is connected to the gripper via the stepper motor link cable as follows.

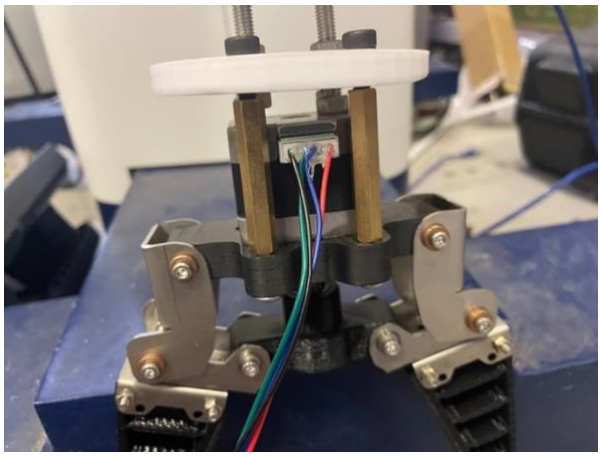


Figure 5-6 Gripper connection

Use SSH to access Raspberry Pi remotely, open the pin control program on Raspberry Pi that controls the opening and closing of the gripper, and wait for the action command to be sent from the host computer. The robot arm is preset to grip the object in ready attitude, gripping attitude, pouring attitude and recovery attitude, the action of this part is fixed action, so the robot arm is programmed directly using the script program.

First let the gripper to the top of the object (p1), and then move to the point to be clamped (p2), at this time the upper computer will receive the 'script finish' by the robot feedback of the script program execution completed sign bit, the upper computer received this signal to the Raspberry Pi to send a signal to close the gripper, the gripper to execute the action. After clamping the object and executing the task action (p3), then put it on the table and open the gripper, execute the double-arm recovery procedure (p5), and the double-arm return to the initial position (p6). The schematic diagram of the whole motion process is shown below.

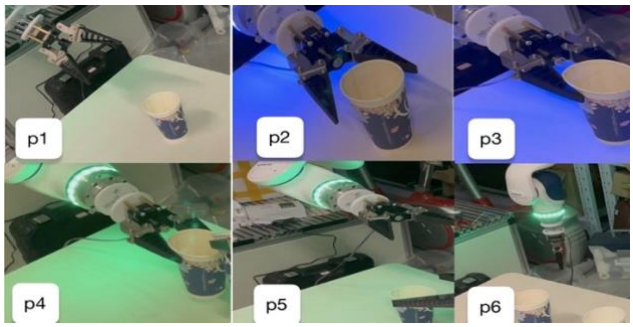


Figure 5-7 Schematic diagram of gripper movement

In the actual experimental scenario, the robot successfully gripped cups, water bottles, milk cartons, and other objects with preset movements. Due to the different

shapes of the objects, it is necessary to set the closing angle of the gripper artificially in order to prevent excessive squeezing of the objects, and here a variety of opening and closing angles are preset as a countermeasure.

5.3 Two-arm collaboration experiment

5.3.1 Real task scenario construction

A coffee cup, which is easily available in life, is used as the recognized object, and one is placed on the left and right side of the real robot.

Fix the RGB-D camera using a tripod at a distance of 1.2 meters horizontally and 1.25 meters vertically from the center of the base of the dual-arm robot (origin of the base scale system). Make the whole tabletop within the field of view of the camera, execute the command in the terminal, move the two cups to the random position, and the double arms will move to the position waiting for grasping and achieve the object grasping at 50% of the maximum speed respectively. The detection scene of the image information can be visualized using the rqt tool, and the depth values of the center of the picture and the center of the bounding box are marked in white numbers in the figure.

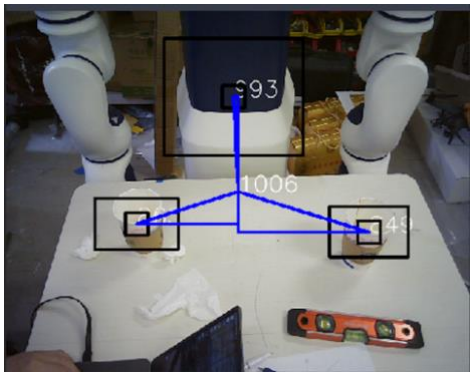


Figure 5-8 Visualization interface of vision system

5.3.2 Hand-eye system performance analysis

Since the 3D positioning algorithm used in this paper ignores the shape of the object itself, the 3D coordinate point of the object positioned is the center of the recognition bounding box, which is usually not the best grasping point. When the relative position of

the camera and the captured object changes, the changing relationship between the positioning point and the best capture point is also changing.

In order to minimize the influence of errors on the experimental results, the grasped object was placed in an approximate area, the 3D coordinate points of the cup output from the recording terminal were observed, and the optimal grasping point was calculated manually. The corresponding parameters in the program were modified to modify the target point of robot motion to the corrected target point.

The experimental steps are shown in the figure below, first open the path planning information and waypoint sending program, then use the 'roslaunch' command to open the launch files of the simulation environment and vision system in turn, and finally run the planning program of the left and right arms, when both arms are planned successfully, the waypoint sending program will send the integrated planning When both arms are successfully planned, the waypoint sending program will send the integrated planning information to the physical robot, and the gripper will execute the opening and closing action when the joint moves to the specified position.

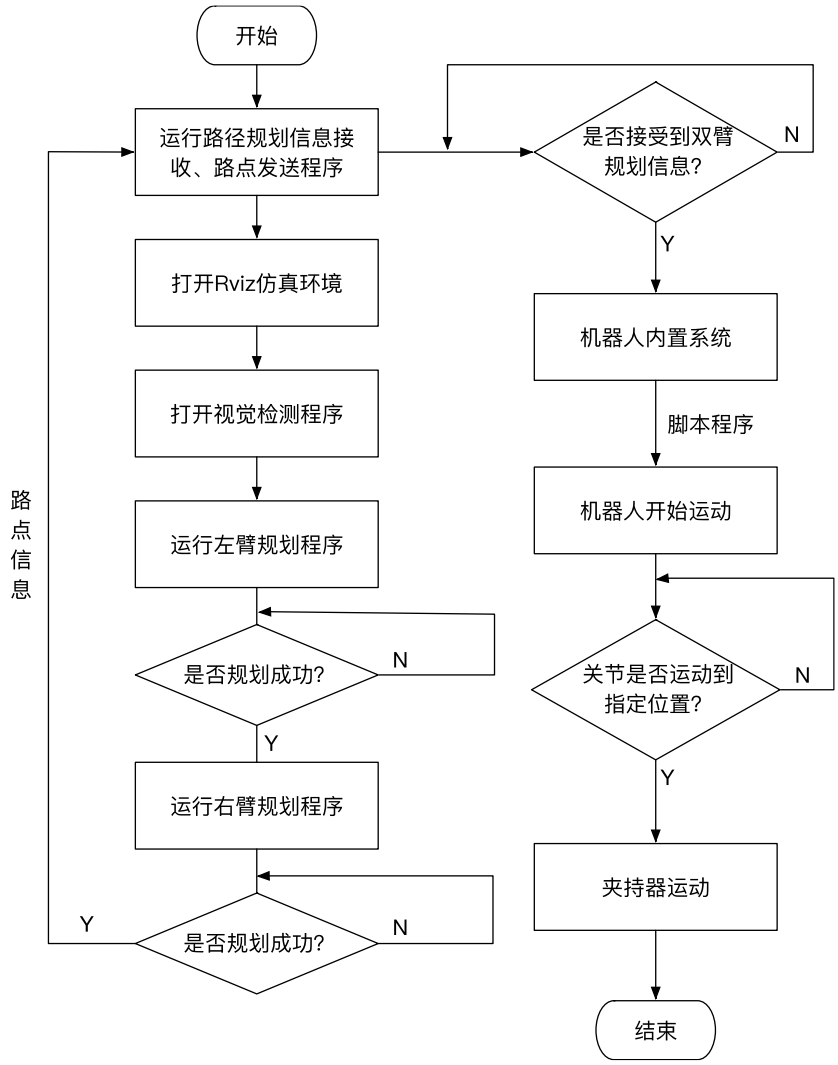


Figure 5-9 Flow chart of program operation

The vision program uses the ROS_INFO function to publish the real-time 3D coordinate points of the left and right cups for user observation in real time. The output information is the 3D coordinate points of the cups under the camera coordinate system and the 3D coordinate points translated to the robot base coordinate system, both in m. The terminal output information is shown in Figure 5-10.

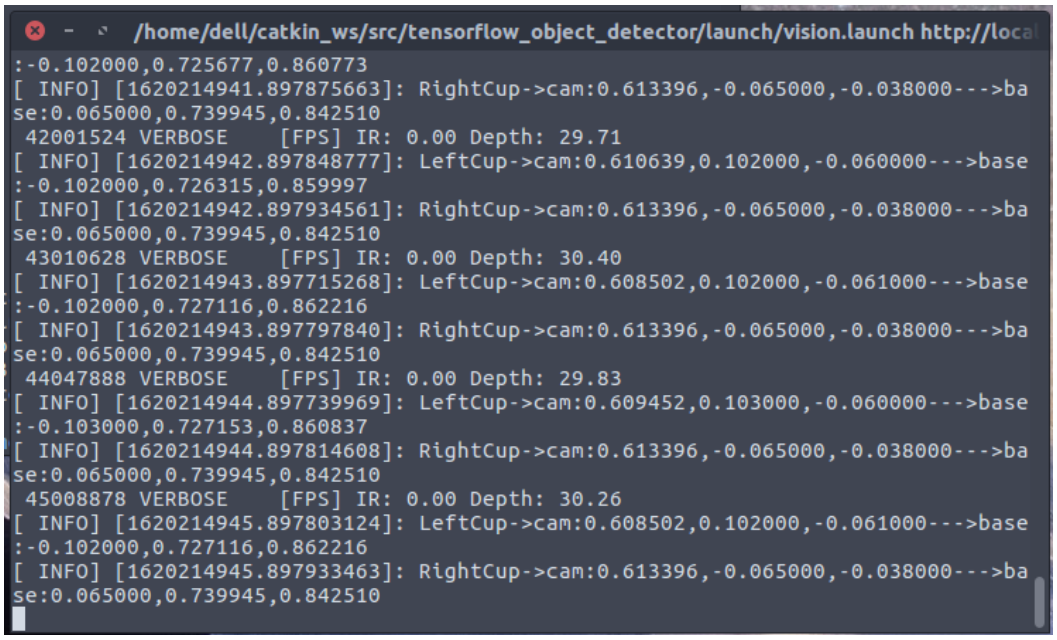


Figure 5-10 Object 3D coordinate information

The planning program will output a success flag after successful planning, and fail to output fail and plan again. The reason for planning failure is that the solution cannot be found, and the terminal will prompt the user to move the cup until the planning is successful or the user terminates (ctrl+c), then the robot will perform the specified task after sending the processed waypoints to the robot via socket.

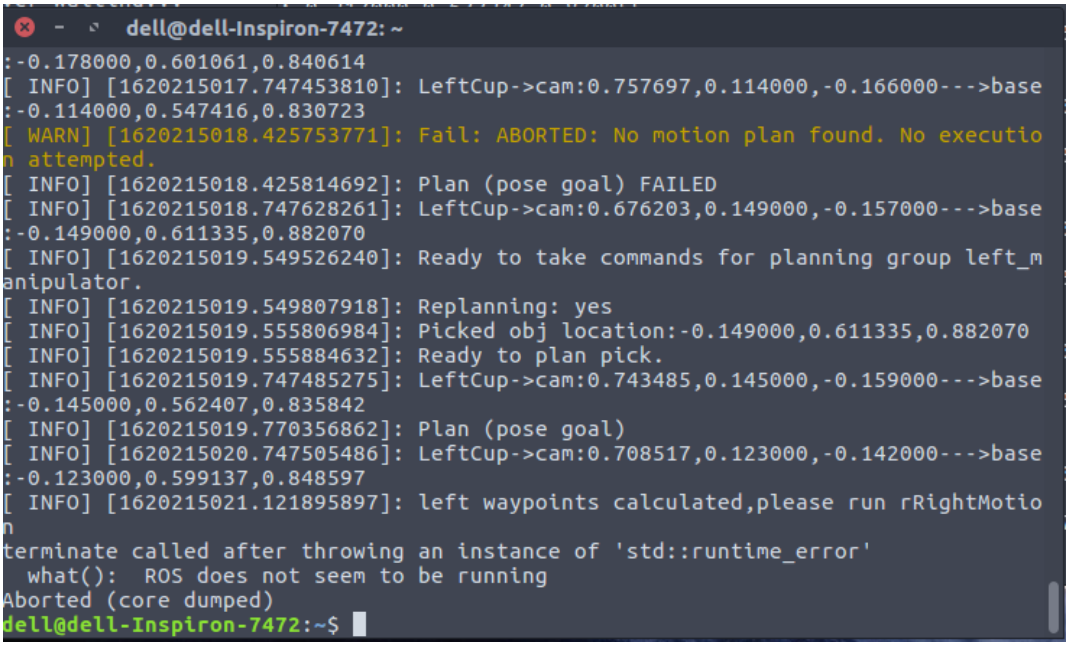


Figure 5-11 Path planning terminal operation information

Use Rviz to visualize the planned path process of the left and right arms from the initial pose to the point to be clamped, as shown in the following figure.

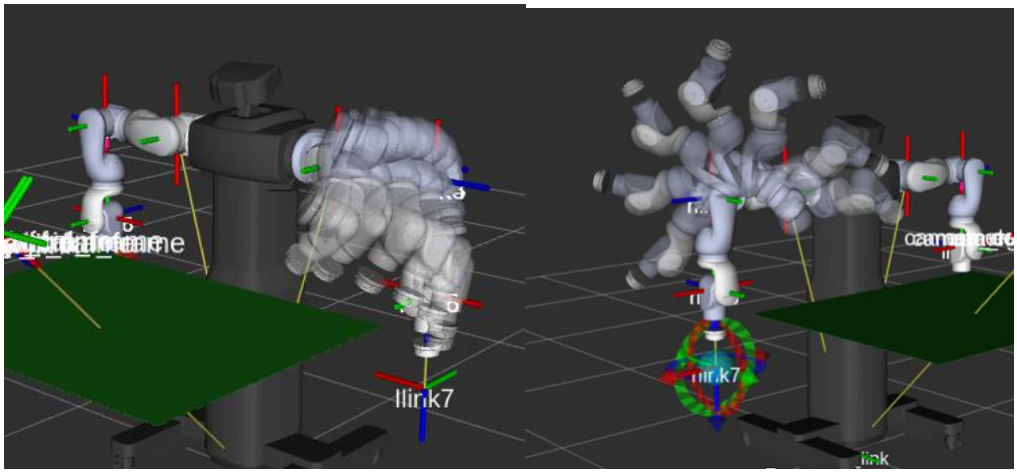


Figure 5-12 Left and right arm trajectory diagram

The motion path of the robot is shown in the following figure. Firstly, the dual-arm robot moves from the initial position to position 1 ready for grasping, and its trajectory is consistent with the planned trajectory in the above figure, and the latter part is to execute the preset fixed action, and the dual-arm robot adjusts the gripper attitude, and then moves to the point to be grasped position 2, and then grasps the cup and executes the pouring action (position 3, 4). Finally, the cup is put down and the initial posture is restored (position 5, 6).

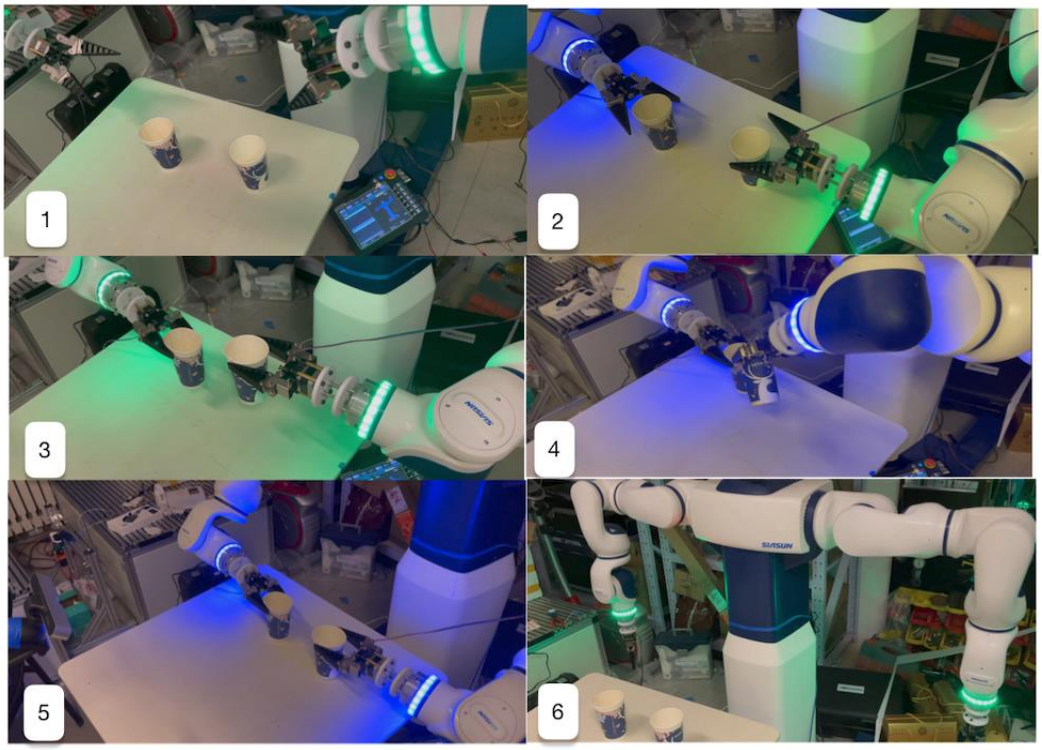


Figure 5-13 Two-arm collaborative water pouring experiment

In 20 experiments, grasping the cup to achieve the pouring action a total of 16 times successfully, of which 4 experiments failed mainly caused by errors in visual positioning, record the terminal output values of XYZ coordinate points in 20 experiments, assuming that points B and C are the geometric center of the cup of the object being clamped, and points A and D are the ideal best clamping points, if the best clamping point and the origin of the coordinate system of the end gripper coincide, it is possible to obtain the ideal clamping effect. In the actual commissioning, the three-dimensional coordinate points of A and B, C and D have the following relationship (unit cm).

$$\begin{cases} |x_A - x_B| = |x_C - x_D| = 5 \\ y_A = y_B = y_C = y_D \\ z_A = z_B = z_C = z_D \end{cases}$$

The four points A, B, C and D are shown schematically in space with the gripper and the cup as follows.

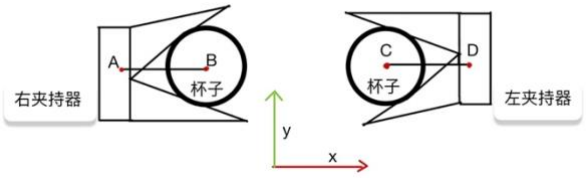


Figure 5-14 Schematic diagram of the best clamping point

The actual running position of the gripper was measured in 20 experiments, and the difference between the ideal gripping position and the ideal gripping position was obtained in the following error statistics graph. In the actual measurement, the 2nd, 7th, 16th and 17th gripper pressed to the edge of the cup causing the clamping failure, and the analysis of the graph below shows that the cause of this phenomenon is the offset of the y coordinate.

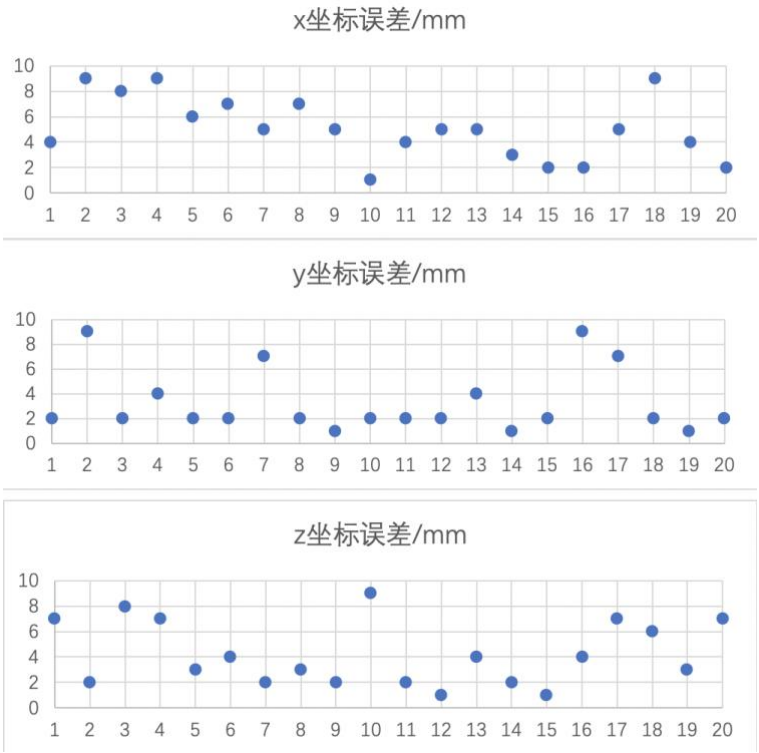


Figure 5-15 Verification point error absolute value

In order to reduce the positioning error, it is necessary to obtain the grasped object morphology and correct the optimal grasping point according to the object morphology. The morphology of the object can be written into the hand-eye system in advance by means of 3D modeling, or the parameters of the object to be grasped can be measured manually in advance, and the grasping point can be corrected by algorithms. Another common method is to use the pcl point cloud library to obtain the point cloud information of the object directly[51] However, due to the high requirements of pcl on the upper processor and camera transmission speed, this system does not use the pcl library.

5.4 Summary of this chapter

This chapter sets up several real environment task environments to test the effectiveness of the hand-eye system, analyzes the advantages and disadvantages of this hand-eye system as well as the causes of task failure lie in the errors of visual localization, and proposes various feasible solutions.

6 Conclusion

6.1 Summary and Outlook

There are the following deficiencies in the visual system.

(1) Since this paper uses the coordinate value of the center of the bounding box as the 3D coordinate point of the object, but this point is not the best grasping point, although the impact of the error is reduced by experimental debugging later, it needs to be modified again if the grasped object is replaced.

(2) There is no design for light sources, and experiments show that light sources have a great influence on the correct recognition rate.

(3) The neural network model structure can be further optimized.

There are the following deficiencies in the intelligent control system.

（1） The attitude of the end-effector of the present system is a preset attitude, and the automatic generation of the end attitude is not possible.

（2） The control lacks real time and is executed once the object coordinate point is collected. If the object is moved, the robot arm will move to the previous 3D coordinate point of the object instead of the current coordinate point.

Based on the above deficiencies, the following improvements are proposed.

（1） The best grip point is corrected by the method of PCL point cloud library.

（2） Design light sources to improve the correct recognition rate.

（3） Use other neural network models to improve the correct recognition rate.

（4） Add obstacle information about non-task objects captured by vision to MoveIt scene planning.

（5） Add deep learning models to automatically generate grasping poses based on object point cloud information.

（6） Use higher performance computers and optimize the code data structure to improve the real-time performance of the system.

6.2 Social and economic analysis

The two-arm robot hand-eye system in this paper can be used in a variety of commercial scenarios, such as coffee making, tea making, wine pouring, etc. Businesses such as cafes and bars can purchase similar robotic systems for service, although the performance of the current system is not enough to replace human labor and the degree of intelligence is very limited, but such robots can be used as a point of attraction to the public and bring commercial value. If the vision system can be optimized, this system can also be used in industrial scenarios.

In this paper, the two-armed robot is provided by the laboratory, the gripper, Raspberry Pi, and depth camera are acquired later, and the performance of Raspberry Pi can be used in more scenarios. The gripper can pick up a wide range of objects, and the depth camera is also used for the hundred dollar camera, which is very economical and scalable.

References

[1] Yin Jinghua. Current situation and future development of industrial robots in China [J]. China New Communication, 2020, 22(20): 135-136.

[2] He Tao. Introduction to the principle and application of machine vision [J]. Technology and Market, 2011, 18(05): 11.

[3] Zhao YQ, Rao Y, Dong SP. A review of deep learning target detection methods [J]. Chinese Journal of Graphical Graphics, 2020, 25(04): 629-654.

[4] Anonymous. ABB introduces a future-oriented human-robot collaboration product: the YuMi dual-arm robot [J]. Automation Technology and Applications, 2014, 33(09): 126.

[5] Park H, Park J, Lee D-H, et al. Compliance-Based Robotic Peg-in-Hole Assembly Strategy Without Force Feedback [J]. IEEE Transactions on Industrial Electronics, 2017, 64(8): 6299-6309.

[6] Chen Xunman. Research on hand-eye vision-based grasping and assembly method for Baxter dual-arm robots with shaft holes [D]. South China University of Technology, 2016, 1-3.

[7] Fuchs M, Borst C, Giordano P R, et al. Rollin' Justin - Design considerations and realization of a mobile platform for a humanoid upper body [C]// International Conference on Robotics and Automation . IEEE, 2009: 2-7.

[8] Manley J E, Halpin S, Radford N, et al. Aquanaut: A New Tool for Subsea Inspection and Intervention [C]// proceedings of the OCEANS 2018 MTS/IEEE Charleston. IEEE, 2018: 1-4.

[9] Coleshill E, Oshinowo L, Rembala R, et al. Dextre: Improving maintenance operations on the international space station [J]. Acta Astronautica, 2009, 64(9-10): 869-874.

[10] Xu F, Zou Fengshan, Zheng Chunhui. Industrial development and application of Xinsong robots [J]. Robotics and Applications, 2011, 000(005): 14-18.

[11] Peng QH, Yang B, Yang ZJ. AeroMEMS: Bionic dual-armed special robot, an all-around war "epidemic" player [J]. Dual-use technology and products, 2020, No.438(04): 20-22.

[12] Umbaugh S E. Computer vision and image processing: a practical approach using cviptools with cdrom [M]. Prentice Hall PTR, 1997.

[13] Sun, Yumeng. Analysis of deep learning applications in computer vision analysis [J]. China New Communication, 2018, 020(023): 169-171.

[14] Xue, Bingxin. Research on key technologies for autonomous navigation and target grasping of mobile robots [D]. Qingdao University of Technology, 2019: 1-3.

[15] Pinto L, Gupta A. Supersizing self-supervision: learning to grasp from 50K tries and 700 robot hours [C]// International Conference on Robotics and Automation. Stockholm, Sweden, IEEE, 2016: 2-3.

[16] Kitagawa S, Wada K, Hasegawa S, et al. Multi-Stage Learning of Selective Dual-Arm Grasping Based on Obtaining and Pruning Grasping Points Through the Robot Experience in the Real World[C]//

References

International Conference on Intelligent Robots and Systems. Madrid, Spain, IEEE, 2018: 1-2.

[17] Mahler J, Liang J, Niyaz S, et al. Dex-net 2.0: deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics [J]. arXiv preprint arXiv:170309312, 2017: 2-3.

[18] Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning [J]. arXiv preprint arXiv:150902971, 2015: 4-6.

[19] Guo Di. Research on target detection and grasping planning for robot operation [D]. Tsinghua University, 2016: 1-3.

[20] Ni Hepeng, Liu Yanan, Zhang Chengrui. Algorithm of Delta robot sorting system based on machine vision [J]. Robotics. 2016. 38(01): 49-55.

[21] Li Chuan-Peng. Research on target recognition and grasping localization based on machine vision and deep learning [D]. North Central University, 2017: 1-3.

[22] Dong J., Chen W. H., Yue H. S.. Automatic identification and localization of tomatoes based on Kinect vision system [J]. Chinese Journal of Agricultural Chemistry, 2014, 35(004): 169-173.

[23] Figueroa J, Contreras L, Pacheco A, et al. Development of an Object Recognition and Location System Using the Microsoft Kinect TM Sensor[C]// Proceedings of the Robot Soccer World Cup. springer, 2011: 440-449.

[24] Songhui M, Mingming S, Chufeng H. Objects detection and location based on mask RCNN and stereo vision[C]// proceedings of the 2019 14th IEEE International Conference on Electronic Measurement & Instruments (ICEMI). IEEE, 2019: 369-373.

[25] Gené-Mola J, Sanz-Cortiella R, Rosell-Polo J R, et al. Fruit detection and 3D location using instance segmentation neural networks and structure-from- motion photogrammetry [J]. Computers and Electronics in Agriculture, 2020: 3-7.

[26]. Hao Jianbao, Cha Jinyan, Xie Lianya. Kinematic modeling and simulation of planar joint robot based on DH algorithm [J]. Mechatronics Engineering Technology, 2018, 47(04): 99-102.

[27] WANG Hongqi, WU Hai-bo, DONG Hao. Design of hand-eye vision servo control system for industrial robots [J]. Sensors and Microsystems. 2019, 38(04): 111-113.

[28] Wang Xuesong, Xu Sifan, Hao Jifei. A new derivation method and solution of the inverse motion equation of MOTOMAN manipulator [J]. Journal of China University of Mining and Technology, 2001, 30(1): 73-76.

[29] Zhou L, Zhou YM, Wang L. Workspace analysis of a two-arm robot based on Monte Carlo method [J]. Mechanical Drives, 2014, 06): 85-87.

[30] Lee S. Dual redundant arm configuration optimization with task-oriented dual arm manipulability [J]. IEEE Transactions on Robotics and Automation, 1989, 5(1): 78-97.

[31]. Liang H, Yu YQ, Zhang CX. A flexible robot dynamics simulation system based on ADAMS and ANSYS [J]. Mechanical Science and Technology, 2002, 21(6): 892-895.

[32] Quigley M, Gerkey B P, Conley K, et al. ROS: An open-source Robot Operating System [J]. 2009.

[33] Abadi, Martin. TensorFlow: Learning Functions at Scale [J]. Acm Sigplan Notices A Monthly Publication of the Special Interest Group on Programming Languages, 2016: 3-7.

References

[34]. Song Shian. Construction and research of a vision sensor-based dual robotic arm collaboration system [D]. Harbin Institute of Technology, 2017: 1-3.

[35]. Liu Jianpo. Research on binocular stereo vision distance measurement system [D]. Xi'an University of Electronic Science and Technology, 2010: 1-3.

[36] Ding JJ, Zhang XD, Gao JJ. A review of research on the application of TOF-based 3D camera [C]// The 12th Youth Conference of the Chinese Society of Instrumentation. Qinhuangdao, 2010: 72-75.

[37] Wang Xiu-Fang, Wang Jiang, Yang Xiang-Dong. Overview of phase laser ranging technology [J]. Laser Journal, 2006, 27(2): 4-5.

[38] Chen Yan-Jun, Zuo Wang-Meng, Wang Kuan-Quan. A review of structured optical coding methods [J]. Small Microcomputer Systems, 2010, 31(9): 1856-1863.

[39] Wang H-C, Huang X-F, Qin Y-Q. Transient fault analysis and processing of wireless positioning system based on TOF ranging [J]. Software, 2011, 32(009): 16-19.

[40] Chen L. Research on Kinect depth image enhancement algorithm [D]. Hunan University, 2013.

[41]. Chi D.X., Wang Y., Ning L.Q., et al. Camera calibration test by Zhang Z.Y. method [J]. Chinese Journal of Agricultural Chemistry, 2015.

[42]. Liu Yan, Li Tengfei. A study on the improvement of Zhang Zhengyou's camera calibration method [J]. Optical Technology, 2014, 40(06): 565-570.

[43]. Zhang YAH. Research on robot grasping system based on Faster R-CNN target detection [D]. University of Chinese Academy of Sciences (Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences), 2019: 1-3.

[44] Wu T, Hu T-B. Stereo vision technology based on monocular ranging [C]// Academic Meeting of the Deep Space Exploration Technology Committee of the Chinese Academy of Astronautics. 2005: 2-3.

[45] Su D, Yu N-M. Deep learning-based algorithm for automatic identification of OBD port occupancy status [J]. Journal of Beijing University of Posts and Telecommunications, 2019, 42(06): 53-61.

[46] Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector [C]// proceedings of the European conference on computer vision. springer, 2016: 21- 37.

[47]. Zhang Hongyao, Li Lun, Zhou Bo, et al. Research on robot hand-eye calibration and its accuracy analysis [J]. Combined machine tools and automated machining technology, 2018, 000(001): 69-72.

[48] Tsai R Y, Lenz R K. A new technique for fully autonomous and efficient 3D robotics hand/eye calibration [J]. IEEE Transactions on Robotics & Automation, 1989, 5(3): 345-358.

[49] T sai R Y, Lenz R. Review of the two-stage camera calibration technique plus some new implementation tips and some new techniques for center and scale calibration [J]. Optical Society of America, Topical Meeting on Machine Vision, 1980: 4-5.

[50] Wu H-X, Shen S-P. Application and theoretical basis of PID control [J]. Control Engineering, 2003, 10(001): 37-42.

[51] Rusu R B, Cousins S. 3D is here: Point Cloud Library (PCL) [C]// proceedings of the IEEE International Conference on Robotics & Automation. IEEE, 2011 IEEE, 2011: 2-3.

Acknowledgements

At this point, my undergraduate career at Beijing Jiaotong University has come to an end. Although I will be studying in a foreign country, the training I received from Jiaotong University will certainly benefit me for the rest of my life and lay a solid foundation for my further study. I will keep in mind the motto of our university, "Knowing and acting in unity", and I will do my best to "drink water and think of the source, love the country and honor the university".

The person I am most grateful to is my graduation design supervisor, who showed me the way forward in my research path like a lighthouse in a fog. My teacher also provided me with many opportunities for practical learning, such as the Beijing UAV competition, through which I deepened my understanding of the ROS system and also succeeded in entering the national competition with the second place in Beijing. When I encountered difficulties in my research, my teacher always took the trouble to help me solve the problems, and I kept communicating at least once a week during the final project.

Thanks to the ROS Q&A community, the foreigners at Stackoverflow for answering many of my technical questions, and to 'Ameyawagh', the user on Github who provided the open source code for ROS Visual Positioning. Without your technical support I might not have been able to finish my graduation design in the time frame. To give back to the open source community for their help, I have uploaded the entire code of the hand-eye system on Github for others to refer to.

In addition, I would like to thank my brothers in the lab, because the system built in this project takes up a lot of space, and the lab was originally compact, I took up the lab space and caused inconvenience to all brothers, so I thank you for your understanding and support.

Finally, I would like to thank my parents and my partner for giving me encouragement and support whenever I was down and feeling sorry for myself.

Appendix

Appendix A Translation of English Literature

SSD: Single Detector

Abstract: We propose a method for detecting objects in images using a single deep neural network. Our method, named SSD, discretizes the output space of bounding boxes into a set of default boxes that have different aspect ratios and sizes at each feature map location. During prediction, the network generates scores for each object class present in each default box, and adjusts the boxes to better match the object shape. In addition, the network combines predictions from multiple feature maps with different resolutions to accommodate the processing of objects of various sizes. Our SSD model is simple compared to approaches that require a regional proposal because it completely eliminates the proposal generation and subsequent pixel or feature resampling stages, and encapsulates all computations in a single network. Experimental results on the PASCALVOC, MS COCO, and ILSVRC datasets confirm that SSD has comparable accuracy and is faster than methods using additional region proposals, while providing a unified framework for training and inference. Compared with other single-stage methods, SSD has better accuracy, even with smaller input image size. For VOC2007, at 300×300 input, SSD achieves 72.1% mAP at 58FPS on Nvidia Titan X and 75.1% mAP at 500×500 input SSD, outperforming similar prior art Faster R-CNN models. Code link: https://github.com/weiliu89/caffe/tree/ssd.

Keywords: real-time object detection; convolutional neural network

1 Introduction

Currently, existing object detection systems are variations of the following approaches: assuming bounding boxes, resampling pixels or features for each box, and applying high-quality classifiers. After the selective search [1] approach, Faster R-CNN [2] achieved leading results in PASCAL VOC, MSCOCO and ILSVRC detection, and this process became a milestone in the field of detection with deeper features, as described in [3]. Although accurate, these methods are too computationally intensive for embedded systems and too slow for real-time or near real-time applications, even for

high-end hardware. The detection speed of these methods is usually measured in frames per second (FPS), with high precision detectors (based on Faster R-CNN) running at as fast as 7 frames per second (FPS). There have been extensive attempts to build faster detectors by studying each stage of the detection process (see related work in Section 4), but so far the significant increase in speed has only come at the cost of a significant decrease in detection accuracy.

This paper presents the first deep network-based object detector that does not resample pixels or features assumed by bounding boxes, but is as accurate as this approach. This results in a significant improvement in the speed of high accuracy detection (72.1% mAP at 58 FPS, 73.2% mAP at 7 FPS for Faster R-CNN, 63.4% mAP at 45 FPS for YOLO) in the VOC2007 test. The fundamental improvement in speed comes from the elimination of the bounding box proposal and the subsequent pixel or feature resampling phase. This is not the first paper to do so (cf [4,5]), but by adding a series of improvements, we managed to improve the accuracy of previous attempts. Our improvements include using separate predictors (filters) with different aspect ratios for detection, predicting object classes and offsets in the bounding box, and applying these filters to multiple feature maps later in the network in order to perform multi-scale detection. With these modifications, we can achieve high accuracy detection using relatively low resolution inputs, further increasing the processing speed. Although these contributions may seem small independently, we note that the resulting system improves the accuracy of high-speed detection of PASCAL VOC from 63.4% mAP for YOLO to 72.1% mAP for our proposed network. this is a large improvement in detection accuracy compared to recent work, and excellent work on residual networks [3]. In addition, the significant increase in the speed of high-quality detection can broaden the range of useful uses of computer vision.

To summarize our contribution as follows.

- We cite SSD, a single detector for multiple classes, which is faster and much more accurate than the single detector of previous techniques (YOLO), and in fact as accurate as slower techniques using regionproposal, pooling (including Faster RCNN)

- The core of the SSD method uses a small convolutional filter to predict the class scores and position offsets of a fixed set of default bounding boxes on the feature map.

- To achieve high detection accuracy, we generate predictions at different scales from feature maps at different scales, and separate the predictions explicitly by aspect ratio.

- In summary, these design features yield simple end-to-end training and high accuracy, further improving the trade-off between speed and accuracy, even with relatively low-resolution image input.

- The experiments include evaluation of model time consumption and accuracy analysis on PASCAL VOC, MS COCO and ILSVRC for different input sizes, and comparison with a range of state-of-the-art methods.

2 Single Detector (SSD)

This section presents our proposed SSD detection architecture (Section 2.1) and the associated training method (Section 2.2). After that, Section 3 presents model details and experimental results for a specific dataset.

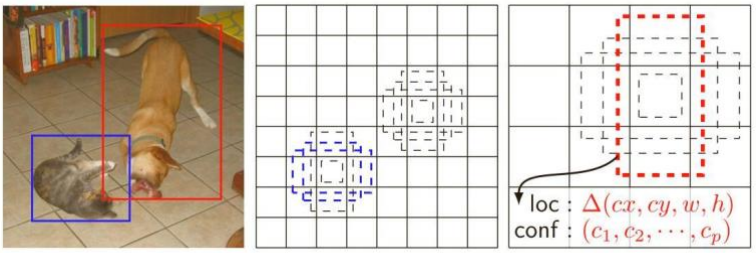


Figure 1 SSD architecture

(a) SSD requires only the input image of each object and the true label frame during training. For convolution processing, we evaluate small sets (e.g., 4) of default boxes with different aspect ratios at each position in several feature maps with different scales (e.g., 8 × 8 and 4 × 4 in (b) and (c)). For each default box, we predict the shape bias for all object classes ((c 1, c2, . . . , cp)). , cp)) for all object classes with shape bias and confidence. During training, we first match these default boxes to the true labeled boxes. For example, two default boxes are matched to cats and dogs, these boxes are positive and the rest are considered negative. The model loss is a weighted sum between the position loss (e.g., smoothed L1 [6]) and the confidence loss (e.g., Softmax).

2.1 Models

The SSD approach is based on a feed-forward convolutional network, which generates a fixed size set of bounding boxes and a score of object classes in the boxes, followed by a non-maximizing suppression step to produce the final detection. The early network is based on a standard architecture for high-quality image classification (truncated before any classification layer*)*, which we refer to as the base network (we used the VGG-16 network as the base in our experiments, and other networks should also produce good results). Then, we added auxiliary structures to the network, yielding detections with the following main features.

Multi-scale feature map detection: We add convolutional feature layers to the end of the truncated base network. These layers are gradually reduced in size to obtain predictions for multi-scale detection. The convolutional model of detection is different for each feature layer (see Overfeat [4] and YOLO [5] for operating on a single-scale feature map).

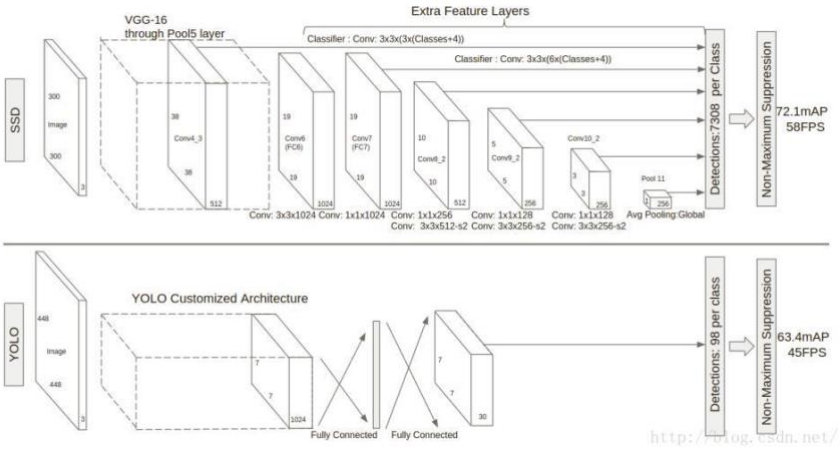


Figure 2 Comparison between two single detection models: SSD and YOLO [5]

Our SSD model adds several feature layers at the end of the base network, which predict the offset of different scales and aspect ratios to the default frame and its associated confidence. The accuracy of the SSD with 300×300 input size in the VOC2007 test is significantly better than that of the YOLO with 448×448 inputs, and also improves the runtime speed, although the YOLO network is faster than VGG16.

Convolutional predictors for detection: Each added feature layer (or optionally an existing feature layer of the underlying network) can use a set of convolutional filters to generate a fixed set of predictions. These are noted at the top of the SSD network architecture in Figure 2. For a feature layer of size m × n with p channels, a 3 × 3 × p convolution kernel convolution operation is used to generate the fraction of categories or coordinate offsets relative to the default box. An output value is generated at each location of size m × n where the convolution kernel operation is applied. The bounding box offset output value is measured relative to the default box, and the default box position is relative to the feature map (see YOLO [5] for an architecture that uses a fully connected layer in the middle instead of the convolution filter used for this step).

Default boxes and aspect ratio: We associate a set of default bounding boxes with each feature map cell of the top-level network. The default boxes perform convolution operations on the feature map so that the position of each box instance is fixed with respect to its corresponding cell. In each feature map cell, we predict the offset relative to the default box shape in the cell, as well as the per-class fraction of instances in each box. Specifically, for each of the k boxes at a given position, we compute the c-class score and four offsets relative to the original default box. This results in a total of (c+4)k filters for each position in the feature map, producing (c+4)kmn outputs for an m×n feature map. For a description of the default boxes, see Fig. 1. Our default boxes are similar to the anchor boxes used in Faster R-CNN [2], but we apply them to feature maps of different resolutions. By using different default box shapes in multiple feature maps, the space of possible output box shapes can be

efficiently discretized.

2.2 Training

The key difference between training SSD and training typical classifiers using region proposal, pooling is that the true label information needs to be assigned to a particular output in a fixed set of detector outputs.The regionproposal phase of Faster R-CNN [2] and MultiBox [7], the YOLO [5] The training phase of YOLO [5] also requires a similar label. Once this designation is determined, the loss function and backpropagation are applied end-to-end. Training also involves the selection of the default set of boxes and scales used for detection, as well as hard negative mining and data augmentation strategies.

Matching strategy: During training, we need to establish the correspondence between the real labels and the default boxes. Note that for each real label box, we select from the default boxes, which vary with position, aspect ratio and scale. At the start, we match each real label box with the best jaccard overlap of the default box. This is the matching method used in the original MultiBox [7], which ensures that each real label box has a matching default box. Unlike MultiBox, matching default boxes with real label jaccard overlap above a threshold (0.5). Adding these matches simplifies the learning problem: it allows the network to predict with high confidence when there are multiple overlapping default boxes, instead of requiring it to choose the one with the largest overlap.

Training: SSD training from MultiBox[7,8], but extended to handle multiple object categories. with denotes the ith default box matching the jth true labeled box of category p. Conversely, according to the matching strategy above, we have 1, meaning that there can be more than one default box matching the jth true labeled box. The overall objective loss function is a weighted sum of position loss (loc) and confidence loss (conf).

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)) \qquad (1)$$

where N is the number of matched default boxes, and the position loss is the smoothed L1 loss between the parameters of the predicted box (l) and the true labeled value box (g) [6]. Similar to Faster R-CNN [2], we regress the center of the bounding box and its width and height offsets. Our confidence loss is a cross-validation of the softmax loss on the multiclass confidence (c) and the weight term α set to 1.

Select the default box scale and aspect ratio: Most convolutional networks reduce the size of the feature map by deepening the number of layers. This not only reduces computational and storage consumption, but also provides some degree of translation and size invariance. To handle different object sizes, some approaches [4,9] suggest converting images to different sizes, then processing each size individually, and then combining the results. However, by using feature maps from several

different layers in a single network for prediction, we can obtain the same results while also sharing parameters across all object scales. Previous studies [10,11] have shown that using feature maps from lower layers can improve the quality of semantic segmentation, since the lower layers capture finer details of the input objects. Similarly, [12] showed that adding global text downsampled from higher-level feature maps can help smooth the segmentation results. Inspired by these approaches, we use both low-level and high-level feature maps for detection prediction. Figure 1 illustrates two example feature maps (8 × 8 and 4 × 4) used in the framework, although in practice, we can use more feature maps with relatively small computational overhead.

It is known that different levels of feature maps in the network have different (empirical) receptive field sizes [13]. Fortunately, within the SSD framework, the default box does not need to correspond to the actual perceptual field at each level. We can design tiling so that location-specific feature maps, learned in response to specific regions of the image and at specific scales of the object. Suppose we want to use m feature maps to make predictions. The scale of the default box for each feature map is calculated as follows.

$$s_k = s_{\min} + \frac{s_{\max} - s_{\min}}{m - 1}(k - 1), \quad k \in [1, m] \tag{2}$$

where smin is 0.2 and smax is 0.95, meaning that the lowest layer has a scale of 0.2, the highest layer has a scale of 0.95, and all layers in between are regularly spaced. We apply different aspect ratios to the default boxes, denoted as ar∈{1,2,3,1/2,1/3}. We can calculate the width and height of each default box. For an aspect ratio of 1, we also add a default box scaled to 1, so that there are 6 default boxes for each feature map position. We set the center of each default box to be, where is the size of the kth square feature map, and then intercept the default box coordinates so that they are always within [0, 1]. In fact, the distribution of default boxes can be designed to best fit a particular data set.

By combining the predictions of all default boxes for many feature maps at all positions with different sizes and aspect ratios, we have a diverse set of predictions that cover a variety of input object sizes and shapes. For example, in Figure 1, the dog is matched to the default box in the 4×4 feature map, but not to any of the default boxes in the 8×8 feature map. This is because those boxes have different scales but do not match the dog's box, and are therefore considered negative samples during training.

Hard negative mining: After the matching step, most of the default frames are negative samples, especially when the number of possible default frames is large. This leads to a significant imbalance of positive and negative samples during training. Instead of using all negative samples, we sorted each

default box using its highest confidence level and selected the first ones so that the ratio between positive and negative samples was at most 3:1. We find that this leads to faster optimization and more stable training.

Data augmentation: In order to make the model more robust to various input object sizes and shapes, each training image is randomly sampled by one of the following options.

- Use the entire original input image

- Sampling a fragment, so that the object has a minimum jaccard overlap of 0.1, 0.3, 0.5, 0.7 or 0.9.

- Randomly sampling a clip

The size of each sampled segment is [0.1,1] of the original image size, and the aspect ratio is between 1/2 and 2. If the center of the true label frame is within the sampled slice, the overlap is preserved. After the above sampling steps, each sampling slice was resized to a fixed size and flipped horizontally with a probability of 0.5.

3 Experimental results

Base network: our experiments are based on the VGG16 [14] network, pre-trained in the ILSVRC CLS-LOC dataset [15]. Similar to DeepLab-LargeFOV [16], we transformed fc6 and fc7 into convolutional layers, sampled parameters from fc6 and fc7, changed pool5 from 2×2-s2 to 3×3-s1, and used atrous algorithm to fill the "holes". We remove all dropout and fc8 layers, and use SGD to fine-tune this model with initial learning rate , 0.9 momentum, 0.0005 weight decay, and batch size 32. The learning rate decay strategy is slightly different for each dataset, we will describe the details later. All training and testing code is written in the caffe framework, open source at https://github.com/weiliu89/caffe/tree/ssd.

3.1 PASCAL VOC2007

On this dataset, we compared Fast R-CNN [6] and Faster R-CNN [2]. All methods use the same training data and pre-trained VGG16 network. In particular, we trained on VOC2007train val and VOC2012 train val (16551images), and tested on VOC2007 (4952images).

Figure 2 shows the architecture details of the SSD300 model. We use conv4_3, conv7 (fc7), conv8_2, conv9_2, conv10_2 and pool11 to predict the position and confidence (for the SSD500 model, conv11_2 is added additionally for prediction), using the "xavier" method The parameters of all newly added convolutional layers are initialized using the "xavier" method [18]. Due to the large size of conv4_3 (38×38), we only place three default boxes on it: a 0.1 scale box and additional boxes with aspect ratios of 1/2 and 2. For all other layers, we set 6 default boxes, as in Section 2.2. As pointed out in [12], since conv4_3 has a different feature scale compared to the other layers, we use the L2

regularization technique introduced in [12] to scale the feature parametrization to 20 at each position in the feature map and learn the scale during backpropagation. We use the learning rate for 40k iterations, then decay it to , and continue training for another 20k iterations. Table 1 shows that our SSD300 model is already more accurate than Fast R-CNN. When training the SSD with a larger 500×500 input image, the results are even more accurate, surprisingly exceeding even the Faster R-CNN by 1.9% mAP.

To understand the performance of our two SSD models in more detail, we used the detection analysis tool from [19]. Figure 3 shows that SSD can detect (large, white areas) various object classes with high quality. It is correct for most of the high confidence detections. The recall is around 85-90%, and much higher than the "weak" (0.1 jaccard overlap) criterion. Compared to R-CNN [20], SSD has less localization error, suggesting that SSD can better localize objects because it directly regresses object shape and classifies object classes instead of using two decoupling steps. However, SSD has more confusion for similar object classes (especially animals), partly because multiple classes share positions.

Method	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast [6]	70.0	77.0	78.1	69.3	59.4	38.3	81.6	78.6	86.7	42.8	78.8	68.9	84.7	82.0	76.6	69.9	31.8	70.1	**74.8**	80.4	70.4
Faster [2]	73.2	76.5	79.0	70.9	**65.5**	**52.1**	83.1	84.7	86.4	52.0	**81.9**	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6
SSD300	72.1	75.2	**79.8**	70.5	62.5	41.3	81.1	80.8	86.4	51.5	74.3	**72.3**	83.5	84.6	80.6	74.5	46.0	71.4	73.8	83.0	69.1
SSD500	**75.1**	**79.8**	79.5	**74.5**	63.4	51.9	**84.9**	**85.6**	**87.2**	**56.6**	80.1	70.0	**85.4**	**84.9**	**80.9**	**78.2**	**49.0**	**78.4**	72.4	**84.6**	**75.5**

Table 1 PASCAL VOC2007 test set detection results

The minimum input image size is 600 for Fast and Faster R-CNNs, and the other settings are the same for both SSD models except for the input image size (300*300 and 500*500). It is obvious that the larger input size gives better results.

Figure 4 shows that SSD is very sensitive to the bounding box size. In other words, it has worse performance for smaller objects than for larger objects. This is not surprising, since small objects may not retain any information at the topmost level. Increasing the input size (e.g., from 300×300 to 500×500) can help improve detection of small objects, but there is still much room for improvement. On the positive side, we can clearly see that SSD performs well on large objects. It is also very robust to different object aspect ratios, as we use default boxes with various aspect ratios for each feature map position.

3.2 Model analysis

To better understand SSD, we also performed several human-controlled experiments to check how each component affects the final performance. For all of the following experiments, we used the exact same setup and input size (300×300), except for variations in components.

	SSD300					
more data augmentation?		✔	✔	✔	✔	✔
use conv4_3?	✔		✔	✔	✔	✔
include $\{\frac{1}{2}, 2\}$ box?	✔	✔		✔	✔	✔
include $\{\frac{1}{3}, 3\}$ box?	✔	✔			✔	✔
use atrous?	✔	✔	✔	✔		✔
VOC2007 `test` mAP	65.4	68.1	69.2	71.2	71.4	72.1

Table 2 Effect of different choices and components on SSD performance

The key data augmentation Fast and Faster R-CNNs are trained using the original images and horizontally flipped (0.5 probability) images. We use a broader sampling strategy, similar to YOLO [5], but it uses photometric distortions that we did not use. Table 2 shows that we can improve mAP by 6.7% with this sampling strategy. we do not know how much our sampling strategy will improve Fast and Faster R-CNNs, but it may not have much effect because they use pooling during classification, which is more robust than artificial settings.

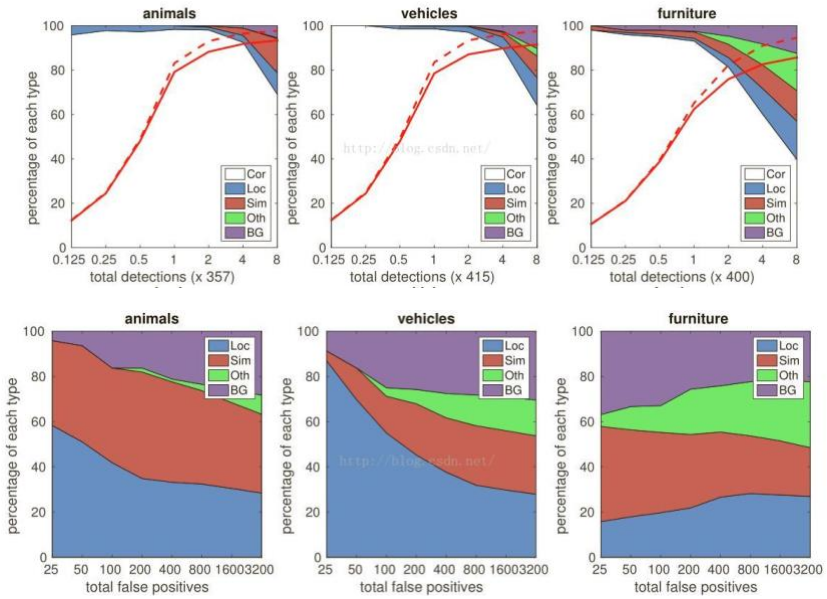


Figure 3 Visualization of SSD 500 on the VOC2007 test set for animal vehicle and furniture performance

The first row shows the cumulative fraction of correct detections (Cor), false positives due to poor localization (Loc), confusion with similar categories (Sim), other categories (Oth), or background (BG). The solid red line reflects the change in recall for the "strong" criterion (0.5 jaccard overlap) as the number of tests increases. The red dashed line uses the "weak" criterion (0.1 jaccard overlap). The bottom row shows the distribution of top-ranked false positive types.

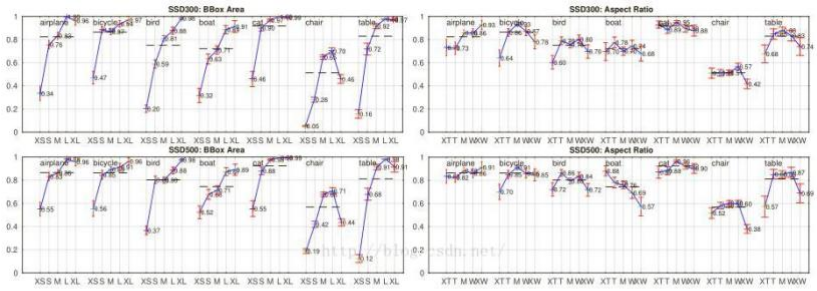


Fig. 4: **Sensitivity and impact of different object characteristics on VOC2007 `test` set.** Each plot shows the normalized AP [19] with standard error bars (red). Black dashed lines indicate overall normalized AP. The plot on the left shows the effects of BBox Area per category, and the right plot shows the effect of Aspect Ratio. Key: BBox Area: XS=extra-small; S=small; M=medium; L=large; XL =extra-large. Aspect Ratio: XT=extra-tall/narrow; T=tall; M=medium; W=wide; XW =extra-wide.

More feature map boosting, Inspired by many semantic segmentation works [10,11,12], we also use the underlying feature map to predict the bounding box output. We compare the model predicted using conv4_3 with the model without it. From Table 2, we can see that by adding conv4_3 for prediction, it has significantly better results (72.1% vs. 68.1%). This is also in line with our intuition that conv4_3 can capture better granularity of objects, especially fine details.

More default box shapes work better, as described in Section 2.2, where by default, 6 default boxes are used per position. If we remove the boxes with 1/3 and 3 aspect ratios, the performance drops by 0.9%. By further removing the boxes with 1/2 and 2 aspect ratios, performance drops by another 2%. Using multiple default box shapes seems to make the network prediction task easier.

Atrous algorithm is better and faster, as described in Section 3, we used the atrous version of VGG16, following DeepLabLargeFOV [16]. If we use the full VGG16, keeping pool5 with 2×2-s2, and without the acquisition parameters from fc6 and fc7, adding conv5_3, the result is slightly worse (0.7%), while the speed is slowed down by about 50%.

3.3 PASCAL VOC2012

Using the same setup as VOC2007, this time, the model was trained with the training validation set of VOC2012, the training validation set of VOC2007, the test set (21503 images), and tested on the test set of VOC2012 (10991 images). With more training data, the model is trained at a learning rate of 60K iterations, and then reduced to a further 20K iterations.

Method	*mAP*	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast [6]	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	**89.3**	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
Faster [2]	70.4	**84.9**	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	**77.1**	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
YOLO [5]	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
SSD300	70.3	84.2	76.3	69.6	53.2	40.8	78.5	73.6	88.0	50.5	73.5	**61.7**	85.8	80.6	81.2	77.5	44.3	73.2	**66.7**	81.1	65.8
SSD500	**73.1**	**84.9**	**82.6**	**74.4**	**55.8**	**50.0**	**80.3**	**78.9**	88.8	**53.7**	76.8	59.4	**87.6**	**83.7**	**82.6**	**81.4**	**47.2**	**75.5**	65.6	**84.3**	**68.1**

Table 3: **PASCAL VOC2012 `test` detection results.** Fast and Faster R-CNN use images with minimum dimension 600, while the image size for YOLO is 448×448.

Table 3 shows the results for the SSD300 and SSD500 models, and we see the same performance trends as we observed in the VOC2007 test. Our SSD300 has outperformed Fast R-CNN, and is very close to Faster R-CNN (only 0.1% difference). By increasing the training and test image size to

500×500, we outperform Faster R-CNN by 2.7%. Compared to YOLO, SSD is significantly better, probably due to the use of convolutional default frames from multiple feature maps and the matching strategy during training.

3.4 MSCOCO

To further validate the SSD architecture, we trained our SSD300 and SSD500 models on the MS COCO dataset. Since the objects in COCO tend to be small, we use smaller default boxes for all layers. We follow the strategy mentioned in Section 2.2, but now our smallest default frame has a scaling of 0.1 instead of 0.2, and the scaling of the default frame on conv4_3 is 0.07 (e.g., corresponding to 21 pixels for a 300×300 image).

We use trainval35k [21] to train our model. Since COCO has more object classes, the gradient is unstable at the beginning. We first trained the model with 4K iterations at a learning rate of 8×, followed by 140K iterations at the learning rate, then 60K iterations at the learning rate, and 40K iterations at the learning rate. Table 4 shows the results on test-dev2015. Similar to what we observed on the PASCAL VOC dataset, SSD300 在 mAP@0.5 和 mAP@[0.5:0.95] outperforms Fast R-CNN, and is close to Faster R-CNN at mAP @ [0.5:0.95]. However, mAP@0.5 更糟, we speculate that this is because the image size is too small, which prevents the model from pinpointing many small objects. By increasing the image size to 500 × 500, our SSD500 outperforms the Faster R-CNN in both criteria. in addition, our SSD500 model is also better than ION [21], which is a multi-size version of the Fast R-CNN, using a recurrent network to explicitly model the context. In Figure 5, we show some detection examples using the SSD500 model in the MSCOCO test-dev.

Method	data	Average Precision		
		0.5	0.75	0.5:0.95
Fast R-CNN [6]	train	35.9	-	19.7
Faster R-CNN [2]	train	42.1	-	21.5
Faster R-CNN [2]	trainval	42.7	-	21.9
ION [21]	train	42.0	23.0	23.0
SSD300	trainval35k	38.0	20.5	20.8
SSD500	trainval35k	43.7	24.7	24.4

Table 4: **MS COCO `test-dev2015` detection results.**

3.5 ILSVRC Preliminary Results

We applied the same network architecture we used for MS COCO to the ILSVRC DET dataset [15]. We used the ILSVRC2014 DET train and val1 to train the SSD300 model, as used in [20]. We first train the model with 4K iterations at a learning rate of 8×, then train the model with 320k iterations at the learning rate, then train the model with 100k iterations and continue with 60k iterations. We can achieve 41.1mAP on the val2 set [20]. Once again, it verifies that SSD is a general framework for high-quality real-time detection.

3.6 Reasoning period

Appendix

Considering the large number of boxes generated from our method, it is necessary to efficiently perform non-maximum suppression (nms) during inference. By using a confidence threshold of 0.01, we can filter out most of the boxes. Then, we use the Thrust CUDA library for sorting, compute the overlap between all remaining boxes using the GPU, apply nms to each class with a jaccard overlap of 0.45, and save the first 200 detections for each image. For SSD300 with 20 VOC classes, this step took about 2.2 ms per image, which is close to the total time spent on all newly added layers.

Table 5 shows the comparison between SSD, Faster R-CNN [2], and YOLO [5].Faster R-CNN uses an extra prediction layer for region proposal and requires feature downsampling. In contrast, our SSD500 method outperforms Faster R-CNN in terms of speed and accuracy. It is worth mentioning that our method SSD300 is the only method that achieves more than 70% mAP in real time. While Faster YOLO [5] can run at 155 FPS, the accuracy is only almost 20% of the mAP.

Method	*mAP*	FPS	# Boxes
Faster R-CNN [2](VGG16)	73.2	7	300
Faster R-CNN [2](ZF)	62.1	17	300
YOLO [5]	63.4	45	98
Fast YOLO [5]	52.7	155	98
SSD300	72.1	58	7308
SSD500	**75.1**	23	20097

Table 5: **Results on Pascal VOC2007 `test`.** SSD300 is the only real-time detection method that can achieve above 70% mAP. By using a larger input image, SSD500 outperforms all methods on accuracy while maintaining a close to real-time speed. The speed of SSD models is measured with batch size of 8.

4 Related Jobs

There are two established methods for object detection in images, one based on sliding windows and the other on regional proposal classification. Before the advent of convolutional neural networks, the two methods used for detection, DeformablePart Model (DPM) [22] and selective search [1], were close in performance. However, after the significant improvements brought by R-CNN [20], which combines selective search region proposal and post classification based on convolutional networks, region proposal object detection methods became common.

The original R-CNN methods have been improved in various ways. The first group of methods improved the quality and speed of post-classification, as it requires classification of thousands of image crops, which is expensive and time-consuming. spPnet [9] has greatly speeded up the original R-CNN methods. It introduces a spatial pyramid pooling layer, which is more robust to region size and scale, and allows the classification layer to reuse feature map features generated at several image resolutions.Fast R-CNN [6] extends SPPnet so that it can fine-tune all layers end-to-end by minimizing the loss of confidence and bounding box regression, which was first introduced in MultiBox [7] for learning objects.

The second group of approaches uses deep neural networks to improve the quality of proposal generation. In recent works, such as MultiBox [7,8], selective search of regional proposal based on low-level image features is replaced by proposal generated directly from a separate deep neural network. This further improves the detection accuracy, but leads to some complex setups that require training two neural networks and the dependencies between them.Faster R-CNN [2] replaces the selective search proposal by a scheme that learns from the region proposal network (RPN), and introduces the integration of the RPN with the FPN by fine-tuning the shared convolutional layers and the prediction layers between the two networks The method of integrating RPN with Fast R-CNN by fine-tuning the shared convolutional layers and the alternation between the prediction layers of both networks. In this way, the regional proposal pools the mid-level feature maps, and the final classification step is faster. Our SSD is very similar to the region proposal network (RPN) in the Faster R-CNN, because we also use fixed (default) frames for prediction, similar to the achor frames in the RPN. However, instead of using these to pool features and evaluate another classifier, we simultaneously generate a score for each object class in each box. Thus, our approach avoids the complexity of merging RPN with Fast R-CNN, and is easier to train and integrate into other tasks.

Another set of methods is directly related to our approach, skipping the proposal step altogether and directly predicting the bounding boxes and confidence levels for multiple categories. OverFeat [4] is a deeper version of the sliding window approach, predicting the bounding box directly from each position of the topmost feature map after knowing the confidence of the underlying object category. yolo [5] uses the entire topmost feature map to predict the confidence of multiple categories and bounding boxes (which are shared by these categories). Our SSD method falls into this category because we do not have a proposal step, but use default boxes. However, our method is more flexible than existing methods because we can use default boxes with different aspect ratios at each feature location in multiple feature maps at different scales. If only one default frame is used at each position of the top-level feature map, our SSD will have an architecture similar to OverFeat [4]; if we use the entire top-level feature map and add a fully connected layer for prediction instead of our convolutional predictor, and do not explicitly consider multiple aspect ratios, we can approximate the reproduction of YOLO [5].

5 Conclusion

This paper presents SSD, a fast single object detector for multiple classes. A key feature of our model is the use of multi-scale convolutional bounding box outputs attached to multiple feature maps at the top of the network. This representation allows us to efficiently model the space of possible frame shapes. We experimentally verify that, given an appropriate training strategy, a larger number of

carefully selected default bounding boxes yields improved performance. We build SSD models that differ by at least an order of magnitude in box prediction location, scale, and aspect ratio compared to existing methods [2,5,7].

We show that given the same VGG-16 infrastructure, SSD outperforms state-of-the-art object detectors in terms of accuracy and speed. Our SSD500 model significantly outperforms the state-of-the-art Faster R-CNN [2] in terms of accuracy for PASCAL VOC and MS COCO, and is three times faster. Our real-time SSD300 model runs at 58 FPS, which is faster than the current real-time YOLO [5], while having significantly higher quality detection.

In addition to its standalone utility, we believe that our complete and relatively simple SSD model provides a great building block for larger systems using object detection components. A promising future direction is to explore it as part of a system using recurrent neural networks for detecting and tracking objects in video.

6 Acknowledgements

This project was started as an internship program at Google, and continued at UNC. We would like to thank Alex Toshev for helpful discussions, and Google's Image Understanding and DistBelief teams. We also thank Philip Amirato and Patrick Polson for their helpful comments. We thank NVIDIA for providing the K40 GPU and thank NSF 1452851 for support.

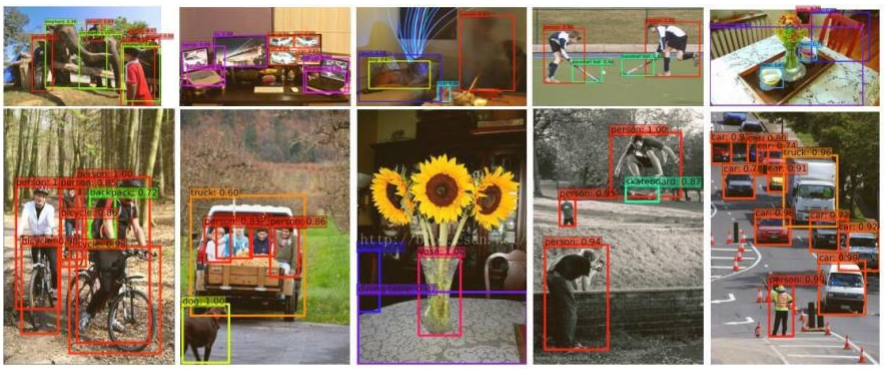


Fig. 5: Detection examples on MS COCO `test-dev` with SSD500 model. We show detections with scores higher than 0.6. Each color corresponds to an object category.

Citation

1. Uijlings, J.R., van de Sande, K.E., Gevers, T.,Smeulders, A.W.: Selective search for object recognition. ijcv (2013)

2. Ren, S., He, K., Girshick, R., Sun, J.: FasterR-CNN: Towards real-time object detection with region proposal networks. in:NIPS.(2015)

3. He, K., Zhang, X., Ren, S., Sun, J.: Deepresidual learning for image recognition. in: CVPR.(2016)

4. Sermanet, P., Eigen, D., Zhang, X., Mathieu, M.,Fergus, R., LeCun, Y.: Overfeat: Integrated recognition, localization anddetection using convolutional networks. in: ICLR.(2014)

5. Redmon, J., Divvala, S., Girshick, R., Farhadi,A.: You only look once: Unified, real-time object detection. in: CVPR.(2016)

6. Girshick,R.: Fast R-CNN. in: ICCV.(2015)

7. Erhan, D.,Szegedy, C., Toshev, A., Anguelov, D.: Scalable object detection using deep neuralnetworks. in: CVPR.(2014)

8. Szegedy, C., Reed, S., Erhan, D., Anguelov, D.: Scalable, high-quality object detection. arXiv preprint arXiv:1412.1441 v3 (2015)

9. He, K.,Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutionalnetworks for visual recognition. in: ECCV.(2014)

10. Long, J., Shelhamer, E., Darrell, T.: Fullyconvolutional networks for semantic segmentation. in: CVPR.(2015)

11. Hariharan, B., Arbeláez, P., Girshick, R.,Malik, J.: Hypercolumns for object segmentation and fine-grained localization.In: CVPR.(2015)

12. Liu, W., Rabinovich, A., Berg, A.C.: ParseNet: looking wider to see better. in: ILCR.(2016)

13. zhou, B., Khosla, A., Lapedriza, A., Oliva, A.,Torralba, A.: Object detectors emerge in deep scene cnns. in: ICLR.(2015)

14. simonyan,K.,Zisserman,A.:Verydeepconvolutionalnetworksforlarge-scaleimagerecog-nition. in: NIPS.(2015)

15. Russakovsky, O., Deng, J., Su, H., Krause, J.,Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg,A.C., Li, F.F.: Imagenet large scale visual recognition challenge. ijcv (2015)

16. Chen, L.C., Papandreou, G., Kokkinos, I.,Murphy, K., Yuille, A.L.: Semantic image segmentation with deep convolutionalnets and fully connected crfs. in: ICLR.(2015)

17. Jia, Y., Shelhamer, E., Donahue, J., Karayev,S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutionalarchitecture for fast feature embedding. in: MM, ACM (2014)

18. Glorot, X., Bengio, Y.: Understanding thedifficulty of training deep feedforward neural networks. in: AISTATS. (2010)

19. Hoiem, D., Chodpathumwan, Y., Dai, Q.: Diagnosing error in object detectors. in: ECCV

2012.(2012)

20. Girshick, R., Donahue, J., Darrell, T., Malik,J.: Rich feature hierarchies for accurate object detection and semanticsegmentation. in: CVPR.(2014)

21. Bell, S., Zitnick, C.L., Bala, K., Girshick, R.: Inside-outside net: detecting objects in context with skip pooling andrecurrent neural networks. In: CVPR.(2016)

22. Felzenszwalb, P., McAllester, D., Ramanan, D.: Adiscriminatively trained, multiscale, deformable part model. in: CVPR. (2008)

SSD: Single Shot MultiBox Detector

Wei Liu1(B), Dragomir Anguelov2, Dumitru Erhan3, Christian Szegedy3, Scott Reed4, Cheng-Yang Fu1, and Alexander C. Berg1

UNC Chapel Hill, Chapel Hill, USA

{wliu,cyfu,aberg}@cs.unc.edu 2 Zoox Inc., Palo Alto, USA drago@zoox.com

Google Inc., Mountain View, USA

{dumitru,szegedy}@google.com

4 University of Michigan, Ann-Arbor, USA reedscot@umich.edu

Abstract. We present a method for detecting objects in images using a single deep neural network. Our approach, named SSD, discretizes the output space of bounding boxes into a set of default boxes over dif- ferent aspect ratios and scales per feature map location. At prediction time, the network generates At prediction time, the network generates scores for the presence of each object cate- gory in each default box and produces adjustments to the box to better match the object shape, SSD is simple relative to methods that require object proposals because it completely eliminates proposal generation and subsequent pixel or feature resampling This makes SSD easy to train and straightforward to integrate into systems that require a Experimental results on the PASCAL VOC, COCO, and ILSVRC datasets confirm that SSD has competitive accuracy to methods that uti lize an additional object proposal step and is much faster, while providing a unified framework for both training and inference. For 300 × 300 input, SSD achieves 74.3 % mAP on VOC2007 test at 59 FPS on a Nvidia Titan X and for 512 × 512 input, SSD achieves 76.9 % mAP, outperforming a comparable state of the art Compared to other single stage methods, SSD has much better accuracy even with a smaller input image size. Code is available at https:/ /github.com/weiliu89/caffe/ tree/ssd.

Appendix

Keywords: Real-time object detection - Convolutional neural network

1 Introduction

Current state-of-the-art object detection systems are variants of the following app- roach: hypothesize bounding boxes, resample pixels or features This pipeline has prevailed on detection bench- marks since the Selective Search work [1] through the current leading results on PASCAL VOC, COCO, and ILSVRC detection all based on Faster R-CNN [2] albeit with deeper features such as [3]. While accurate, these approaches have been too computationally intensive for embedded systems and, even with high- end hardware, too slow for real-time Often detection speed for these approaches is measured in frames per second, and even the fastest high-accuracy detector, Faster R-CNN, The detection speed for these approaches is measured in frames per second, and even the fastest high-accuracy detector, Faster R-CNN, operates at only 7 frames per second (FPS). There have been many attempts to build faster detectors by attacking each stage of the detec- tion pipeline (see related work in Sect. 4), but so far, Significantly increased speed comes only at the cost of significantly decreased detection accuracy.

This paper presents the first deep network based object detector that does not resample pixels or features for bounding box hypotheses and is as accurate This results in a significant improvement in speed for high-accuracy detection (59FPS with mAP 74.3% on VOC2007 test, vs Faster R- CNN 7FPS with mAP 73.2% or YOLO 45FPS with mAP 63.4%). The fundamental improvement in speed comes from eliminating bounding box proposals and the subsequent pixel or feature resampling stage. first to do this (cf. [4,5]), but by adding a series of improvements, we manage to increase the accuracy significantly over previous attempts. improvements include using a small convolutional filter to predict object categories and offsets in bounding box locations, using separate predictors (filters) for different aspect ratio detections, and applying these filters to multiple feature maps from the later stages of a network in order to With these modifications-especially using multiple layers for prediction at different scales-we can achieve high-accuracy detection at multiple scales. -We can achieve high-accuracy using relatively low resolution input, further increasing detection speed. seem small independently, we note that the resulting system improves accuracy on real-time detection for PASCAL VOC from 63.4% mAP for YOLO to 74.3% mAP This is a larger relative improvement in detection accuracy than that from the recent, very high-profile work on residual networks [3]. Furthermore, significantly improving the speed of high-quality detection can broaden the range of settings where computer vision is useful.

We summarize our contributions as follows:

- - We introduce SSD, a single-shot detector for multiple categories that is faster than the previous state-of-the-art for single shot detectors (YOLO), and significantly more accurate, in fact as accurate as slower techniques that perform explicit region proposals and pooling (including Faster R-CNN).
- - The core of SSD is predicting category scores and box offsets for a fixed set of default bounding boxes using small convolutional filters applied to feature maps.
- - To achieve high detection accuracy we produce predictions of different scales from feature maps of different scales, and explicitly separate predictions by aspect ratio.

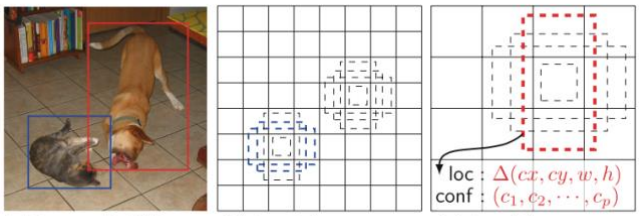


Fig. 1. SSD framework. (a) SSD only needs an input image and ground truth boxes for each object during training. In a convolutional fashion, we evaluate a small set (e.g. 4) of default boxes of different aspect ratios at each location in several feature maps with different scales (e.g. 8×8 and 4×4 in (b) and (c)). For each default box, we predict both the shape offsets and the confidences for all object categories ((c1, c2, - - - , cp)). At training time, we first match these default boxes to the ground truth boxes. For example, we have matched two default boxes with the cat and one with the The model loss is a weighted sum between localization loss (e.g. Smooth L1 [6]) and confidence

These design features lead to simple end-to-end training and high accuracy, even on low resolution input images, further improving the speed vs accuracy Experiments include timing and accuracy analysis on models with varying input size evaluated on PASCAL VOC, COCO, and ILSVRC and are com- pared to a range of recent state-of-the-art approaches.

2 The Single Shot Detector (SSD)

This section describes our proposed SSD framework for detection (Sect. 2.1) and the associated training methodology (Sect. 2.2). Afterwards, Sect. 3 presents dataset-specific model details and experimental results.

2.1 Model

The SSD approach is based on a feed-forward convolutional network that pro- duces a fixed-size collection of bounding boxes and scores for the presence of The early network layers are based on a The early network layers are based on a standard architecture used for high quality image classification (truncated before any classification layers), which we will call the base network1. We then add auxiliary structure to the network to produce detections with the following key features:

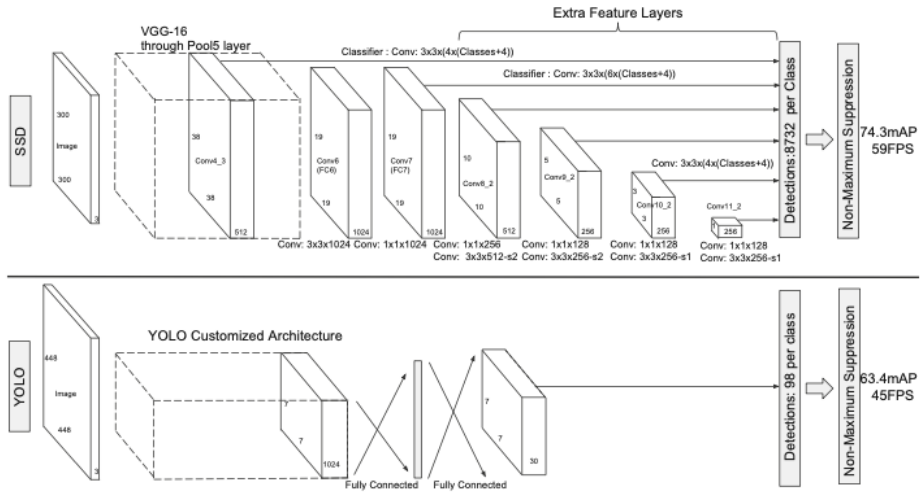


Fig. 2. A comparison between two single shot detection models: SSD and YOLO [5]. Our SSD model adds several feature layers to the end of a base network, which predict the offsets to default boxes of different scales and aspect ratios and their associated confidences. Our SSD model adds several feature layers to the end of a base network, which predict the offsets to default boxes of different scales and aspect ratios and their associated confidences. SSD with a 300 × 300 input size significantly outperforms its 448 × 448 YOLO counterpart in accuracy on VOC2007 test while also improving the speed.

We add convolutional feature layers to the end of the truncated base network. These layers decrease in size progres- sively and allow predictions of detections at multiple scales**.** The convolutional model for predicting detections is different for each feature layer (cf Overfeat [4] and YOLO [5] that operate on a single scale feature map).

Each added feature layer (or optionally an existing feature layer from the base network) can produce a fixed set of detection predictions using a set of convolutional filters**.** These are indi- cated on top of the SSD network architecture in Fig.2. For a feature layer of size m × n with p channels, the basic element for predicting parameters of a potential detection is a 3 × 3 × p small kernel that produces either a score for a At each of the m × n locations where the kernel is applied, it produces an output value. The bounding box offset output values are measured relative to a default box position relative to each feature map location (cf the architecture of YOLO [5] that uses an intermediate fully connected layer instead of a convolutional filter for this step).

We associate a set of default bounding boxes with each feature map cell, for multiple feature maps at the top of the network The default boxes tile the feature map in a convolutional manner, so that the position of each box relative to its corresponding cell is fixed. feature map cell, we predict the offsets relative to the default box shapes in the cell, as well as the per-class scores that indicate the presence of a class Specifically, for each box out of k at a given location, we compute c class scores and the 4 offsets relative to the original This results in a total of (c + 4)k filters that are applied around each location in the feature map, yielding (c + 4)kmn outputs for a m × n For an illustration of default boxes, please refer to Fig. 1. Our default boxes are similar to the anchor boxes used in Faster R-CNN [2],

Our default boxes are similar to the anchor boxes used in Faster R-CNN [2], however we apply them to several feature maps of different resolutions. Allowing different default box shapes in several feature maps let us efficiently discretize the space of possible output box shapes.

2.2 Training

The key difference between training SSD and training a typical detector that uses region proposals, is that ground truth information needs to be assigned to specific outputs in the fixed set of detector outputs. Some version of this is also required for training in YOLO [5] and for the region proposal stage of Faster R- CNN [2] and MultiBox [7]. Once this assignment is determined, the loss function and back propagation are applied end-to-end. Training also involves choosing the set of default boxes and scales for detection as well as the hard negative mining and data augmentation strategies.

During training we need to determine which default boxes correspond to a ground truth detection and train the network accordingly**.** For each ground truth box we are selecting from default boxes that vary over location, aspect ratio, and scale. We begin by matching each ground truth box to the default box with the best Jaccard overlap (as in MultiBox [7]). Unlike MultiBox, we then match default boxes to any ground truth with Jaccard overlap higher than a threshold (0.5). This simplifies the learning problem, allowing the network to predict high scores for multiple overlapping default boxes rather than requiring it to pick only the one with maximum overlap.

Training Objective. The SSD training objective is derived from the MultiBox objective [7,8] but is extended to handle multiple object categories. let x_{pij} = {1, 0} be an indicator for matching the i-th default box to the j-th ground truth box of category p. In the matching strategy above, we can have $\sum_i x_{ij}^p \geq 1$. .The overall objective loss function is a weighted sum of the localization loss (loc) and the confidence loss (conf):

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)) \qquad (1)$$

where N is the number of matched default boxes, and the localization loss is the Smooth L1 loss [6] between the predicted box (l) and the ground truth box (g) parameters. Similar to Faster R-CNN [2], we regress to offsets for the center of the bounding box and for its width and height. our confidence loss is the softmax loss over multiple classes confidences (c) and the weight term α is set to 1 by cross validation.

Choosing Scales and Aspect Ratios for Default Boxes. To handle differ- ent object scales, some methods [4,9] suggest processing the image at different However, by utilizing feature maps from several different layers in a single network for prediction we can mimic the same effect, while also sharing parameters across all object scales. Previous works [10,11] have shown that using feature maps from the lower Previous works [10,11] have shown that using feature maps from the lower layers can improve semantic segmentation quality because the lower layers capture more fine details of the

input objects. Similarly, [12] showed that adding global context pooled from a feature map can help smooth the segmentation results. Figure 1 shows two exemplar feature maps (8 × 8 and 4 × 4) which are used in the framework. In practice, we can use many more with In practice, we can use many more with small computational overhead.

We design the tiling of default boxes so that specific feature maps learn to be responsive to particular scales of the objects. The scale of the default boxes for each feature map is computed as:

$$s_k = s_{\min} + \frac{s_{\max} - s_{\min}}{m - 1}(k - 1), \quad k \in [1, m] \qquad (2)$$

By combining predictions for all default boxes with different scales and aspect ratios from all locations of many feature maps, we have a diverse set of For example, in Fig. 1, the dog is matched to a default box in the 4 × 4 feature map, but not to This is because those boxes have different scales and do not match the dog box, and therefore are considered as This is because those boxes have different scales and do not match the dog box, and therefore are considered as negatives during training.

Hard Negative Mining. After the matching step, most of the default boxes are negatives, especially when the number of possible default boxes is large. This introduces a significant imbalance between the positive and negative training examples. Instead of using all the negative examples, we sort them Instead of using all the negative examples, we sort them using the highest confidence loss for each default box and pick the top ones so that the ratio between the negatives and positives is at most 3:1. that this leads to faster optimization and a more stable training.

To make the model more robust to various input object sizes and shapes, each training image is randomly sampled by one of the following options:

- Use the entire original input image.
- Sample a patch so that the minimum Jaccard overlap with the objects is 0.1, 0.3, 0.5, 0.7, , or 0.9. Randomly sample a patch.

The size of each sampled patch is [0.1, 1] of the original image size, and the aspect ratio is between 12 and 2. We keep the overlapped part of the ground truth We keep the overlapped part of the ground truth box if the center of it is in the sampled patch. After the aforementioned sampling step, each sampled patch is resized to fixed size and is horizontally After the aforementioned sampling step, each sampled patch is resized to fixed size and is horizontally flipped with probability of 0.5, in addition to applying some photo-metric distortions similar to those described in [13].

3 Experimental Results

Base Network. Our experiments are all based on VGG16 [14], which is pre-trained on the ILSVRC CLS-LOC dataset [15]. Similar to DeepLab- LargeFOV [16], we convert fc6 and fc7 to convolutional layers, subsample para- meters from fc6 and fc7, change pool5 from 2×2 -s2 to 3×3-

s1, and use the atrous algorithm to fill the "holes". We remove all the dropout layers and the fc8 layer. we fine-tune the resulting model using SGD with initial learning rate 10^{-3}, 0.9 momentum, We fine-tune the resulting model using SGD with initial learning rate 10-3, 0.9 momentum, 0.0005 weight decay, and batch size 32. The learning rate decay policy is slightly different for each dataset, and we will describe details later. training and testing code is built on Caffe [17] and is open source at https://github.com/weiliu89/caffe/tree/ssd.

3.1 PASCAL VOC2007

On this dataset, we compare against Fast R-CNN [6] and Faster R-CNN [2] on VOC2007 test (4952 images). All methods use the same pre-trained VGG16 network.

Figure 2 shows the architectural details of the SSD300 model. we use conv4 3, conv7 (fc7), conv8 2, conv9 2, conv10 2, and conv11 2 to predict both location and confidences2. We initialize the parameters for all the newly added convolu- tional layers with the "xavier" method [18]. For conv4 3, conv10 2 and conv11 2, we only associate 4 default boxes at each feature map location - omitting aspect ratios of $_{13}$ and 3. For all Since, as pointed out in [12], conv4 3 has a different feature scale com- pared to the other Since, as pointed out in [12], conv4 3 has a different feature scale com- pared to the other layers, we use the L2 normalization technique introduced in [12] to scale the feature norm at each location in the feature map to 20 and learn the scale We use the 10^{-3} learning rate for 40k

Table 1. PASCAL VOC2007 test detection results. both Fast and Faster R- CNN use input images whose minimum dimension is 600. the two SSD models have exactly The two SSD models have exactly the same settings except that they have different input sizes (300 × 300 vs. 512 × 512). It is obvious that larger input size leads to better results, and more data always helps. data: "07": VOC2007 trainval, "07+12": union of VOC2007 and VOC2012 trainval. "07+12+COCO": first train on COCO trainval35k then fine- tune on 07+12.

Method	data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast [6]	07	66.9	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8
Fast [6]	07+12	70.0	77.0	78.1	69.3	59.4	38.3	81.6	78.6	86.7	42.8	78.8	68.9	84.7	82.0	76.6	69.9	31.8	70.1	74.8	80.4	70.4
Faster [2]	07	69.9	70.0	80.6	70.1	57.3	49.9	78.2	80.4	82.0	52.2	75.3	67.2	80.3	79.8	75.0	76.3	39.1	68.3	67.3	81.1	67.6
Faster [2]	07+12	73.2	76.5	79.0	70.9	65.5	52.1	83.1	84.7	86.4	52.0	81.9	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6
Faster [2]	07+12+COCO	78.8	84.3	82.0	77.7	68.9	65.7	88.1	88.4	88.9	63.6	86.3	70.8	85.9	87.6	80.1	82.3	53.6	80.4	75.8	86.6	78.9
SSD300	07	68.0	73.4	77.5	64.1	59.0	38.9	75.2	80.8	78.5	46.0	67.8	69.2	76.6	82.1	77.0	72.5	41.2	64.2	69.1	78.0	68.5
SSD300	07+12	74.1	74.6	80.2	72.2	66.2	47.1	82.9	83.4	86.1	54.4	78.5	73.9	84.4	84.5	82.4	76.1	48.6	74.3	75.0	84.3	74.0
SSD300	07+12+COCO	79.6	80.9	86.3	79.0	**76.2**	57.6	87.3	88.2	88.6	60.5	85.4	**76.7**	**87.5**	**89.2**	84.5	81.4	55.0	81.9	**81.5**	85.9	78.9
SSD512	07	71.6	75.1	81.4	69.8	60.8	46.3	82.6	84.7	84.1	48.5	75.0	67.4	82.3	83.9	79.4	76.6	44.9	69.9	69.1	78.1	71.8
SSD512	07+12	76.8	82.4	84.7	78.4	73.8	53.2	86.2	87.5	86.0	57.8	83.1	70.2	84.9	85.2	83.9	79.7	50.3	77.9	73.9	82.5	75.3
SSD512	07+12+COCO	**81.5**	**86.9**	**87.5**	**82.0**	75.5	**66.4**	**88.2**	**88.7**	**89.3**	**65.2**	**88.3**	74.4	87.1	88.9	**85.9**	**84.5**	**57.6**	**84.6**	80.7	**87.1**	**81.7**

Iterations, then we continue training for 10k iterations with 10^{-4} and 10^{-5}. When we train SSD on a larger 512 × 512 input image it is even more accurate, If we train SSD with more (i.e. 07 + 12) data, we observe that SSD300 is already better than Faster R-CNN by 0.9 % and that If we take models

trained on COCO trainval35k as described in Sect. 3.4 and fine-tuning them on the 07 + 12 dataset with SSD512, we achieve the best results: 81.5 % mAP.

To understand the performance of our two SSD models in more details, we used the detection analysis tool from [19]. Figure3 shows that SSD can detect various object categories with high quality (large white area). The recall is around 85-90 %, and is much higher with "weak "Compared to R-CNN [20], SSD has less localization error, indicating that SSD can localize objects better because it directly learns to regress the object shape and classify object categories instead of using two decoupled steps. However, SSD has more confusions with similar object categories (especially for animals), partly because we share locations for multiple categories. In other words, it has much worse performance on smaller objects than bigger objects. Increasing the input size (e.g. from 300 × 300 to 512 × 512) can help On the positive side, we can clearly see that SSD performs really well on large And it is very robust to different object aspect ratios because we use default boxes of various aspect ratios per feature map location.

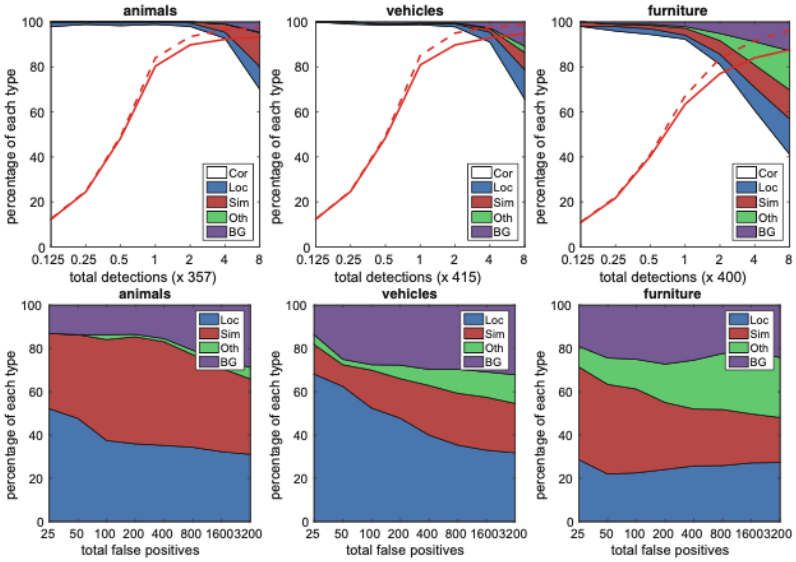


Fig. 3. Visualization of performance for SSD 512 on animals, vehicles, and furniture from VOC2007 test using [19]. The top row shows the cumulative frac- tion of detections that are correct (Cor) or false positive due to poor localization (Loc), confusion with similar categories (Sim), with others (Oth), or with background (BG). The bottom row shows the distribution of top-ranked false positive types.

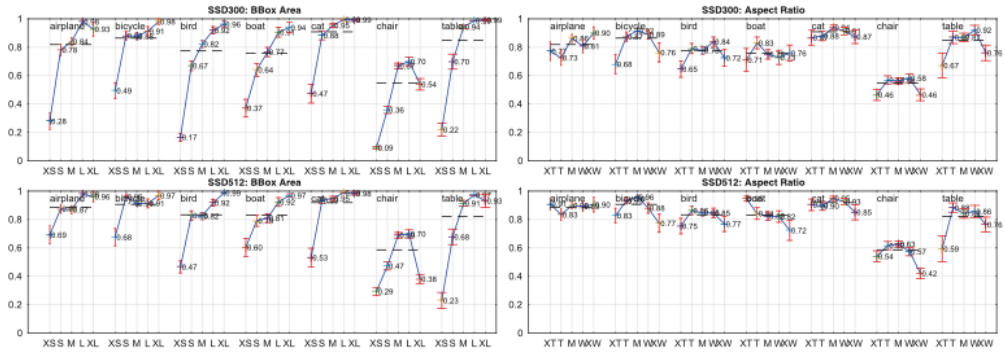


Fig. 4. Sensitivity and impact of different object characteristics on VOC2007 test set using [19]. The plot on the left shows the effects of BBox Area per category, and the right plot shows the effect of Aspect Ratio.

3.2 Model Analysis

To understand SSD better, we carried out controlled experiments to examine how each component affects performance. For all the experiments, we use the same settings and input size (300 × 300), except for specified changes to the settings or component(s). For all the experiments, we use the same settings and input size (300 × 300), except for specified changes to the settings or component(s).

Table 2. Effects of various design choices and components on SSD performance.

	SSD300				
more data augmentation?		✔	✔	✔	✔
include $\{\frac{1}{2}, 2\}$ box?	✔		✔	✔	✔
include $\{\frac{1}{3}, 3\}$ box?	✔			✔	✔
use atrous?	✔	✔	✔		✔
VOC2007 `test` mAP	65.5	71.6	73.7	74.4	74.3

Table 3. Effects of multiple layers.

Source layers from:						mAP use boundary boxes?		# Boxes
conv4_3	conv7	conv8_2	conv9_2	conv10_2	conv11_2	Yes	No	
✔	✔	✔	✔	✔	✔	74.3	63.4	8732
✔	✔	✔	✔	✔		74.6	63.1	8764
✔	✔	✔	✔			73.8	68.4	8942
✔	✔	✔				70.7	69.2	9864
✔	✔					64.2	64.4	9025
	✔					62.4	64.0	8664

Data Augmentation is Crucial. Fast and Faster R-CNN use the original image and the horizontal flip to train. we use a more extensive sampling strategy, similar to YOLO [5]. Table 2 shows that we can improve 8.8 % mAP with this sampling strategy. We do not know how much our sampling strategy will benefit Fast and Faster R-CNN, but they are likely to benefit less because they they are likely to benefit less because they use a feature pooling step during classification that is relatively robust to object translation by design.

As described in Sect. 2.2, by default we use 6 default boxes per location. , the performance drops by 0.6%. By further removing the boxes with $_{12}$ and 2 aspect ratios, the performance drops another 2.1%. Using a variety of default box shapes seems to make the task of predicting boxes easier for the network.

Atrous is Faster. As described in Sect. 3, we used the atrous version of a sub-sampled VGG16, following DeepLab-LargeFOV [16]. If we use the full VGG16, keeping pool5 with 2 × 2 - s2 and not subsampling parameters from fc6 and fc7, and add conv5 3 for prediction, the result is about the same while the speed is about 20 % slower.

Multiple Output Layers at Different Resolutions is Better. A major contribution of SSD is using default boxes of different scales on different output To measure the advantage gained, we progressively remove layers and compare results. For a fair comparison, every time we remove a layer, we adjust the default box tiling to keep the total number of boxes similar to the original (8732). This is done by stacking more scales of boxes on remaining layers and adjusting scales of boxes if needed. Table 3 shows a decrease in accuracy with fewer layers, dropping monotonically from 74.3 to 62.4. When we stack boxes of multiple scales on a We tried the strategy used in Faster R-CNN [2], ignoring boxes which are on the We observe some interesting trends. For example, it hurts the performance by a large margin if we use very coarse feature maps (e.g. conv11 2 (1 × 1) or conv10 2 (3 × 3)). The reason might be that we do not have enough large boxes to cover large objects after the pruning. When we use primarily finer resolution maps, the performance starts increasing again because even after pruning a sufficient number of large boxes remains. performance is the worst, reinforcing the message that it is critical to spread boxes of different scales over different layers.

3.3 PASCAL VOC2012

We use the same settings as those used for our basic VOC2007 experiments above, except that we use VOC2012 trainval and VOC2007 trainval and test (21503 images) for training, and test on VOC2012 test (10991 images). images) for training, and test on VOC2012 test (10991 images). We train the models with $^{10\text{-}3}$ learning rate for 60k iterations, then $^{10\text{-}4}$ for 20k iterations. Table 4 shows the results of our We see the same performance trend as we observed on VOC2007 test. Our SSD300 improves accuracy over Fast/Faster R-CNN. increasing the training and test- ing image size to 512 × 512, we are 4.5 % more accurate than Faster R-CNN. accurate, likely due to the use of convolutional default boxes from multiple feature maps and our matching strategy during training.

from models trained on COCO, our SSD512 achieves 80.0 % mAP, which is 4.1 % higher than Faster R-CNN.

3.4 COCO

To further validate the SSD framework, we trained our SSD300 and SSD512 architectures on the COCO dataset. Since objects in COCO tend to be smaller than We follow the strategy mentioned in Sect. 2.2, but now our smallest default box has a scale of 0.15 We follow the strategy mentioned in Sect. 2.2, but now our smallest default box has a scale of 0.15 instead of 0.2, and the scale of the default box on conv4 3 is 0.07 (e.g. 21 pixels for a 300 × 300 image).

We use the trainval35k [21] for training. We first train the model with 10^{-3} learning rate for 160k iterations, and then continue training for 40k iter- ations with 10^{-4} and 40k iterations with 10^{-5}.

Table5 shows the results on

Table 4. pASCAL VOC2012 test detection results. fast and Faster R-CNN use images with minimum dimension 600, while the image size for YOLO is 448 × 448. data : "07++12": union of VOC2007 trainval and test and VOC2012 trainval. "07++12+COCO": first train "07++12+COCO": first train on COCO trainval35k then fine-tune on 07++12.

Method	data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast[6]	07++12	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
Faster[2]	07++12	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
Faster[2]	07++12+COCO	75.9	87.4	83.6	76.8	62.9	59.6	81.9	82.0	91.3	54.9	82.6	59.0	89.0	85.5	84.7	84.1	52.2	78.9	65.5	85.4	70.2
YOLO[5]	07++12	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
SSD300	07++12	72.4	85.6	80.1	70.5	57.6	46.2	79.4	76.1	89.2	53.0	77.0	60.8	87.0	83.1	82.3	79.4	45.9	75.9	69.5	81.9	67.5
SSD300	07++12+COCO	77.5	90.2	83.3	76.3	63.0	53.6	83.8	82.8	92.0	59.7	82.7	63.5	89.3	87.6	85.9	84.3	52.6	82.5	**74.1**	88.4	74.2
SSD512	07++12	74.9	87.4	82.3	75.8	59.0	52.6	81.7	81.5	90.0	55.4	79.0	59.8	88.4	84.3	84.7	83.3	50.2	78.0	66.3	86.3	72.0
SSD512	07++12+COCO	**80.0**	**90.7**	**86.8**	**80.5**	**67.8**	**60.8**	**86.3**	**85.5**	**93.5**	**63.2**	**85.7**	**64.4**	**90.9**	**89.0**	**88.9**	**86.8**	**57.2**	**85.1**	72.8	**88.4**	**75.9**

Table 5. COCO `test-dev2015` detection results.

Method	Data	Mean average precision		
		0.5	0.75	0.5:0.95
Fast R-CNN [6]	train	35.9	-	19.7
Fast R-CNN [21]	train	39.9	20.5	19.4
Faster R-CNN [2]	train	42.1	-	21.5
Faster R-CNN [2]	trainval	42.7	-	21.9
Faster R-CNN [22]	trainval	45.3	24.2	23.5
ION [21]	train	42.0	23.0	23.0
SSD300	trainval35k	41.2	23.2	23.4
SSD512	trainval35k	**46.4**	**26.7**	**27.7**

test-dev2015. Similar to what we observed on the PASCAL VOC dataset, SSD300 is better than Fast R-CNN in both mAP@0.5 and mAP@[0.5:0.95]. SSD300 has a similar mAP@[0.5:0.95] to Faster R-CNN [22]. However, the mAP@0.5 is worse and we conjecture that it is because the image size is too small, which prevents the model from detecting many small objects. But over- all, SSD can localize objects more accurately. By increasing the image size to 512×512, our SSD512 is better than Faster R-CNN in both criteria. In addition, our SSD512 model is also better than ION [21], a multi-scale version of Fast R- CNN with explicit modeling of context using a recurrent network . In Fig. 5, we show some detection examples on COCO test-dev with the SSD512 model.

3.5 Preliminary ILSVRC Results

We applied the same network architecture we used for COCO to the ILSVRC DET dataset [15]. We train a SSD300 model using the ILSVRC2014 DET train and val1 as used in [20]. We first train the model with 10^{-3} learning rate for 320k iterations, and then continue training for 80k iterations with 10^{-4} and 40k.

3.6 Inference Time

Considering the large number of boxes generated from our method, it is essen- tial to perform non-maximum suppression (nms) efficiently during By using a confidence threshold of 0.01, we can filter out most boxes. We then apply nms with Jaccard overlap of 0.45 per class and keep the top 200 This step costs about 1.7 ms per image for SSD300 and 20 VOC classes, which is close to the total time (2.4 ms) spent on all newly added layers.

Table6 shows the comparison between SSD, Faster R-CNN [2], and YOLO [5]. Both our SSD300 and SSD512 method outperforms Faster R-CNN in both speed and accuracy. Although Fast YOLO [5] can run at 155 FPS, it has lower accuracy by To the best of our knowledge, SSD300 is the first real-time method to achieve above 70% mAP. Note that about 80% of the forward time is spent on the base network (VGG16 in our case). Therefore, using a faster base network could further improve the speed, making the SSD512 model real-time as well.

Table 6. Results on Pascal VOC2007 `test`. SSD300 is the only real-time detection method that can achieve above 70 % mAP. By using a larger input image, SSD512 outperforms all methods on accuracy while maintaining a close to real-time speed. Using a test batch size of 8 improves the speed further.

Method	*mAP*	FPS	Test batch size	# Boxes
Faster R-CNN [2] (VGG16)	73.2	7	1	300
Faster R-CNN [2] (ZF)	62.1	17	1	300
YOLO [5]	63.4	45	1	98
Fast YOLO [5]	52.7	155	1	98
SSD300	74.3	46	1	8732
SSD512	76.8	19	1	24564
SSD300	74.3	59	8	8732
SSD512	76.8	22	8	24564

4 Related Work

There are two established classes of methods for object detection in images, one based on sliding windows and the other based on region proposal Classification. Before the advent of convolutional neural networks, the state of the art for those two approaches - Deformable Part Model (However, after the dramatic improvement brought on by R-CNN [20], which combines selective search region proposals and con- volutional network based post-classification, region proposal object detection methods became prevalent.

The original R-CNN approach has been improved in a variety of ways. The first set of approaches improve the quality and speed of post-classification, The first set of approaches improve the quality and speed of post-classification, since it requires the classification of thousands of image crops, which is expensive and time-consuming. It introduces a spatial pyramid pooling layer that is more robust to region size and scale and allows the classification layers Fast R-CNN [6] extends SPPnet so that it can fine-tune all layers end-to-end by minimizing a loss for both confidences and bounding box regression, which was first introduced in MultiBox [7] for learning objectness.

The second set of approaches improve the quality of proposal generation using deep neural networks. In the most recent works like MultiBox [7,8], the Selective Search region proposals, which are based on low-level image features, are replaced by proposals generated directly from a separate deep This further improves the detection accuracy but results in a somewhat complex setup, requiring the training of two neural networks with a dependency between them. Faster R-CNN [2] replaces selective search proposals by ones learned from a region proposal network (RPN), and introduces a method to integrate the RPN with Fast R-CNN. Faster R-CNN [2] replaces selective search proposals by ones learned from a region proposal network (RPN), and introduces a method to integrate the RPN with Fast R-CNN by alternating between fine-tuning shared convolutional layers and prediction layers for these two networks This way region proposals are used to pool mid-level features and the final classification step is less expensive. proposal network (RPN) in Faster R-CNN in that we also use a fixed set of (default) boxes for prediction, similar to the achor boxes in the

RPN. using these to pool features and evaluate another classifier, we simultaneously produce a score for each object category in each box. Thus, our approach avoids the complication of merging RPN with Fast R-CNN and is easier to train, faster, and straightforward to integrate in other tasks.

Another set of methods, which are directly related to our approach, skip the proposal step altogether and predict bounding boxes and confidences for OverFeat [4], a deep version of the sliding window method, predicts a bounding box directly from each location of the YOLO [5] uses the whole topmost feature map to predict both confidences for multiple categories and bounding boxes (which are shared for these categories). Our SSD method falls in this category because we do not have the proposal step but use the default boxes. existing methods because we can use default boxes of different aspect ratios on each feature location from multiple feature maps at different scales. we only use one default box per location from the topmost feature map, our SSD would have similar architecture to OverFeat [4]; if we use the whole topmost feature map and add a fully connected layer for predictions instead of our convolutional

Fig. 5. Detection examples on COCO test-dev with SSD512 model. we show detections with scores higher than 0.6. each color corresponds to an object Each color corresponds to an object category.

predictors, and do not explicitly consider multiple aspect ratios, we can approx- imately reproduce YOLO [5].

5 Conclusions

Appendix

This paper introduces SSD, a fast single-shot object detector for multiple cate- gories. A key feature of our model is the use of multi-scale A key feature of our model is the use of multi-scale bounding box outputs attached to multiple feature maps at the top of the network. We experimentally validate that given appropriate training strategies, a larger num- ber of carefully chosen default bounding boxes results in improved performance. We experimentally validate that given appropriate training strategies, a larger num- ber of carefully chosen default bounding boxes results in improved performance, and aspect ratio, than existing methods [5,7].

We demonstrate that given the same VGG-16 base architecture, SSD com- pares favorably to its state-of-the-art object detector counterparts in terms of Our SSD512 model significantly outperforms the state-of-the-art Faster R-CNN [2] in terms of accuracy on PASCAL VOC and COCO, Our real time SSD300 model runs at 59FPS, which is faster than the current real time YOLO [5] alternative, while producing markedly superior detection accuracy.

Apart from its standalone utility, we believe that our monolithic and rel- atively simple SSD model provides a useful building block for larger systems A promising future direction is to explore its use as part of a system using recurrent neural networks to detect and track objects in video simultaneously. and track objects in video simultaneously.

This work was started as an internship project at Google and continued at UNC. We would like to thank Alex Toshev for helpful discussions and are indebted to the Image Understanding and DistBelief teams at Google. We would like to thank Alex Toshev for helpful discussions and are indebted to the Image Understanding and DistBelief teams at Google. thank NVIDIA for providing GPUs and acknowledge support from NSF 1452851, 1446631, 1526367, 1533771.

References

1. Uijlings, J.R., van de Sande, K.E., Gevers, T., Smeulders, A.W.: Selective search for object recognition. iJCV 104, 154 (2013)
2. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. in: NIPS (2015)
3. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. in: CVPR (2016)
4. Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y.: Overfeat: integrated recognition, localization and detection using convolutional networks. in: ICLR (2014)
5. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. in: CVPR (2016)
6. Girshick, R.: Fast R-CNN. in: ICCV (2015)
7. Erhan, D., Szegedy, C., Toshev, A., Anguelov, D.: Scalable object detection using

deep neural networks. in: CVPR (2014)

8. Szegedy, C., Reed, S., Erhan, D., Anguelov, D.: Scalable, high-quality object detec-

tion. arXiv preprint v3 (2015). arXiv:1412.1441

9. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional

networks for visual recognition. in: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. lNCS, vol. 8691, pp. 346-361. Springer, Heidelberg (2014). Doi:10.1007/978-3-319-10578-9 23

10. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. in: CVPR (2015)
11. Hariharan, B., Arbel´aez, P., Girshick, R., Malik, J.: Hypercolumns for object seg- mentation and fine-grained localization. in: CVPR (2015)
12. Liu, W., Rabinovich, A., Berg, A.C.: ParseNet: looking wider to see better. in: ILCR (2016)
13. Howard, A.G.: Some improvements on deep convolutional neural network based image classification. arXiv preprint (2013). arXiv:1312.5402
14. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. in: NIPS (2015)

SSD: Single Shot MultiBox Detector 37

15. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: Imagenet large scale visual recognition challenge. ijcv 115, 211 (2015)
16. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Semantic image segmentation with deep convolutional nets and fully connected CRFs. In: ICLR (2015)
17. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: convolutional architecture for fast feature embedding. in: MM. ACM (2014)
18. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. in: AISTATS (2010)
19. Hoiem, D., Chodpathumwan, Y., Dai, Q.: Diagnosing error in object detectors. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7574, pp. 340-353. Springer, Heidelberg (2012). Doi:10.1007/ 978-3-642-33712-3 25
20. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accu- rate object detection and semantic segmentation. in: CVPR (2014)
21. Bell, S., Zitnick, C.L., Bala, K., Girshick, R.: Inside-outside net: detecting objects in context with skip pooling and recurrent neural networks. in: CVPR (2016)
22. COCO:Common Objects in Context (2016). http://mscoco.org/dataset/#

detections-leaderboard. accessed 25 July 2016

Appendix B Waypoint Sending Procedures

```
# Waypoint Sender
import time
import socket
import string

class Position:
 def __init__(self,x,y,z):
        self.x=x
        self.y=y
        self.z=z

class PosePo(Position):
 def __init__(self,x,y,z):
        super(). __init__(self,x,y,z,Rx,Ry,RZ)
        self.Rx=Rx
        self.Ry=Ry
        self.Rz=Rz



def init_socket():
 #send cmd to the dual-arm robot
 debug_local=0
 if debug_local==1:
        target_ip1=("127.0.0.1",7000)
```

```
else :
        target_ip1=("192.168.1.50",2000)
 #create socket target
 global sk
 sk = socket.socket()
 sk.connect(target_ip1)

def remove_np(s):
 return ''.join( c for c in s if c in string.printable)

def check_status():
 while True:
        msg=sk.recv(1000)
        msg=msg.decode()
        print(msg)
        if msg.find("finish")! =-1:
                try:
                        target_ip_tool=("192.168.0.10",7000)
                        tsk=socket.socket()
                        tsk.connect(target_ip_tool)
                        cmd="executeC"
                        tsk.send(cmd.encode())
                except:
                        print("cannot connect pi")
                Finally:
                        print(cmd)
                        break
 return True

        #print(counter)
 #print("motion done")

def main():
 #define motion for robot
```

```
try:
    init_socket()
except:
    print("fail to connect robot")
with open("wayPoints.txt",'r') as f:
    script1=f.read()
    script1=remove_np(script1)
    f.close()
sk.send(script1.encode())
status=check_status()
#rotate cup and place
script2="<start>LP/ movej_pose([-340,640,1040,78,51,90],40,20-1);
sync_flag(1);movej_pose([-162,670,1220,131,24,107],40,20,-1) ; sync_flag(2);movej_pose([-
300,620,870,78,51,90],40,20,-1);/RP/ movej_pose([313,620,1017,90,40,-95],40,20,-
1);sync_flag(1);sync_ flag(2); movej_pose([400,620,870,90,40,-95],40,20,-1);<end>"
#back home
script3="<start>LP/ sleep(6000);movej_pose([-162,620,1200,78,51,90],40,20,-
1);movej_pose([-900,700,1250,90,0,167],40,20,-1); movej([-0,-0,-0,-90,0,-0,0],30,2000,-1);
/RP/sleep(6000); movej_pose([313,620,1200,90,40,-95],40,20,-1); movej_pose([900,700,1250,90,
0,167],40,20,-1);movej([0,-0,0,90,-0,-0,0],30,2000,-1);<end>"
if status:
    sk.send(script2.encode())
    while True:
        flag=sk.recv(1000)
        flag=flag.decode()
        print(flag)
        if flag.find("finish")! =-1:
            sk.send(script3.encode())
            try:
                target_ip_tool=("192.168.0.10",7000)
                tsk=socket.socket()
                tsk.connect(target_ip_tool)
                cmd="executeO"
                tsk.send(cmd.encode())
```

```
        except:
            print("fail to connect pi")
        Finally:
            break

if __name__=='__main__':
 main()
```

Appendix C Visual Positioning Procedure

```
#! /usr/bin/env python
## Author: Rohit
## Date: July, 25, 2017
# Purpose: Ros node to detect objects using tensorflow

import os
import sys
import cv2
import numpy as np
import tf as rostransform
import tf.msg as rostransform_msg
import geometry_msgs.msg
import math
try:
    import tensorflow as tf
except ImportError:
    print("unable to import TensorFlow. Is it installed?")
    print(" sudo apt install python-pip")
    print(" sudo pip install tensorflow ")
    sys.exit(1)
```

```
# ROS related imports
import rospy
from std_msgs.msg import String , Header
from sensor_msgs.msg import Image
from cv_bridge import CvBridge, CvBridgeError
from vision_msgs.msg import Detection2D, Detection2DArray, ObjectHypothesisWithPose, ObjInfo
import message_filters

# Object detection module imports
import object_detection
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as vis_util



DEBUG_MODE=1 # 1=verbos 0=not_verbose
#

#
PUBLISH_IN_CAMERA_FRAME=False
#
IMAGE_HEIGHT=480
IMAGE_WIDTH=640
#
rospy.set_param('zinverse_depth_scaling')



# SET FRACTION OF GPU YOU WANT TO USE HERE
GPU_FRACTION = 0.4

######### Set model here ############
MODEL_NAME = 'ssd_mobilenet_v1_coco_11_06_2017'
# By default models are stored in data/models/
MODEL_PATH = os.path.join(os.path.dirname(sys.path[0]),'data','models' , MODEL_NAME)
```

```
# This is the actual model that is used for the object detection.
PATH_TO_CKPT = MODEL_PATH + '/frozen_inference_graph.pb'
######### Set the label map file here ###########
LABEL_NAME = 'mscoco_label_map.pbtxt'
# By default label maps are stored in data/labels/
PATH_TO_LABELS = os.path.join(os.path.dirname(sys.path[0]),'data','labels', LABEL_NAME)
######### Set the number of classes here #########
NUM_CLASSES = 90
llocdata=np.array([])
rlocdata=np.array([])

detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')

## Loading label map
# Label maps map indices to category names, so that when our convolution network predicts `5`,
# we know that this corresponds to `airplane`.  Here we use internal utility functions,
# but anything that returns a dictionary mapping integers to appropriate string labels would be
fine
label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories = label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=NUM_CLASSES, use_display_name=True)
category_index = label_map_util.create_category_index(categories)



# Setting the GPU options to use fraction of gpu that has been set
config = tf.ConfigProto()
config.gpu_options.per_process_gpu_memory_fraction = GPU_FRACTION
```

```
# Detection

class Detector:

    def __init__(self):

        self.bridge = CvBridge()
        self.sess = tf.Session(graph=detection_graph,config=config)

        #self.depth_image_sub = messag e_filters.Subscriber('image', Image)
        #self.color_image_sub = message_filters.Subscriber('/camera/rgb/image_raw', Image)
        self.depth_image_sub = message_filters.Subscriber('/camera/depth/image_rect_raw',
Image)
        self.color_image_sub = message_filters.Subscriber('/camera/rgb/image_raw', Image)
        ApproximateTimeSynchronizer([self.depth_image_sub, self.color_image_sub],
10,0.1,allow_headerless=True)
        ts.registerCallback(self.processing_cb)

        self.tf_listener = rostransform.TransformListener()

        self.image_pub = rospy.Publisher("debug_image",Image, queue_size=1)
        self.left_obj_pub = rospy.Publisher("LeftObjects", ObjInfo, queue_size=1)
        self.right_obj_pub = rospy.Publisher("RightObjects", ObjInfo, queue_size=1)
        self.object_pub = rospy.Publisher("objects", Detection2DArray, queue_size=1)
        #using rostransform instead of tf for namespace reasons
        self.pub_tf = rospy.Publisher("/tf", rostransform_msg.tfMessage, queue_size=1)
        self.test_pub = rospy.Publisher("new_debug_image",Image, queue_size=1)
        self.debug_pub = rospy.Publisher("depth_debug_image",Image, queue_size=1)
```

```
def processing_cb(self, depth_img_data, color_img_data):

    #print('----')

    data = color_img_data # for consistency with prev source

    objArray = Detection2DArray()
    try:
        cv_image = self.bridge.imgmsg_to_cv2(data, "bgr8")
        depth_img_data.encoding = "mono16" # need this to work
        depth_image = self.bridge.imgmsg_to_cv2(depth_img_data, "mono16")
        #print('cv_image shape: '+str(cv_image.shape))
        #print('depth_image shape: '+str(depth_image.shape))
    except CvBridgeError as e:
        print(e)



    image=cv2.cvtColor(cv_image,cv2.COLOR_BGR2RGB)

    # the array based representation of the image will be used later in order to prepare the
    # result image with boxes and labels on it.
    image_np = np.asarray(image)
    # Expand dimensions since the model expects images to have shape: [1, None, None,
3]
    image_np_expanded = np.expand_dims(image_np, axis=0)
    image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')
    # Each box represents a part of the image where a particular object was detected.
    boxes = detection_graph.get_tensor_by_name('detection_boxes:0')
    # Each score represents how level of confidence for each of the objects.
    # Score is shown on the result image, together with the class label.
    scores = detection_graph.get_tensor_by_name('detection_scores:0')
    classes = detection_graph.get_tensor_by_name('detection_classes:0')
    num_detections = detection_graph.get_tensor_by_name('num_detections:0')
```

```
            (boxes, scores, classes, num_detections) = self.sess.run([boxes, scores, classes,
num_detections],
                feed_dict={image_tensor: image_np_expanded})

            objects=vis_util.visualize_boxes_and_labels_on_image_array(
                image,
                np.squeeze(boxes),
                np.squeeze(classes).astype(np.int32),
                np.squeeze(scores),
                category_index,
                use_normalized_coordinates=True,
                line_thickness=2)

            objArray.detections = []
            objArray.header=data.header
            object_count=1


            for i in range(len(objects)):
                object_count+=1
                obj_struct = self.object_predict(objects[i],data.header,image_np,cv_image)
                objArray.detections.append(obj_struct)

        #print(objects[i])

                # get depth of object and write debug info on cv_image
                avg_depth, center, cv_image = self.get_object_depth(obj_struct, cv_image,
depth_image, i)
                if avg_depth == 0:
                    continue

                # calculate 3D tf and publish it
                obj_id = objects[i][0]
                obj_category = category_index[obj_id]['name']
```

```
                self.object_tf_publisher(center, avg_depth, obj_category, i)


            self.object_pub.publish(objArray)


            if DEBUG_MODE == 1:

                # /debug_image - base tensorflow image detection
                img=cv2.cvtColor(image_np, cv2.COLOR_BGR2RGB)
                image_out = Image()
                try:
                    image_out = self.bridge.cv2_to_imgmsg(img, "bgr8")
                except CvBridgeError as e:
                    print(e)
                image_out.header = data.header
                self.image_pub.publish(image_out)

                depth_image_out = depth_image # TODO: remove this debug image and all instances hereafter
                try:
                    cv_image_out = self.bridge.cv2_to_imgmsg(cv_image, "bgr8")
                    depth_image_out = self.bridge.cv2_to_imgmsg(depth_image_out, "mono16")
                except CvBridgeError as e:
                    print(e)
                self.test_pub.publish(cv_image_out)
                self.debug_pub.publish(depth_image_out)


        def object_tf_publisher(self, center, avg_depth, obj_category, i):

            # (col,row) for rgb image
```

```
u = center[1] - IMAGE_HEIGHT/2
v = center[0] - IMAGE_WIDTH/2
# u/v = y/z, so need a sclaing factor
y_scaling_factor = rospy.get_param('zinverse_depth_scaling/y')
z_scaling_factor = rospy.get_param('zinverse_depth_scaling/y')
y = y_scaling_factor * v
z = z_scaling_factor * u
try:
    x = math.sqrt(avg_depth**2 - y**2 - z**2)
except:
    print('[ERROR] calculating x')
    return # was continue in that loop

if DEBUG_MODE == 1:
    Picked_ObjName='cup'
    global llocdata
    global rlocdata
    # print('b_Object: '+obj_category+'_'+str(i))
    # print('Pixel coords: '+str(u)+','+str(v))
    # print('3D Coords : '+str(x)+','+str(y)+','+str(z))
    if obj_category.find(Picked_ObjName)! =-1:
        if y>0:
            objInfo=ObjInfo()
            objInfo.name=obj_category
            objInfo.x=x
            objInfo.y=y
            objInfo.z=z
            #print("leftCup find")
            self.left_obj_pub.publish(objInfo)

        else:
            objInfo=ObjInfo()
            objInfo.name=obj_category
            objInfo.x=x
```

```
                objInfo.y=y
                objInfo.z=z
                #print("rightCup find")
                self.right_obj_pub.publish(objInfo)
        # if llocdata.size==12:
        # llocdata=llocdata.reshape(4,3)
        # mean_arr=np.mean(llocdata,axis=0)
        # std_arr=np.std(llocdata, axis=0)
        # mean_x=mean_arr[0]
        # std_x=std_arr[0]
        # range_low = mean_x - std_x
        # range_high = mean_x+std_x
        # data_x=llocdata[:,0]
        # index=np.where([data_x<range_low,data_x>range_high])
        # llocdata=np.delete(llocdata,index,axis=0)
        # position_arr=np.mean(llocdata,axis=0)
        # x=position_arr[0]
        # y=position_arr[1]
        # z=position_arr[2]


        # if rlocdata.size==12:
        # rlocdata=rlocdata.reshape(4,3)
        # mean_arr=np.mean(rlocdata,axis=0)
        # std_arr=np.std(rlocdata, axis=0)
        # mean_x=mean_arr[0]
        # std_x=std_arr[0]
        # range_low = mean_x - std_x
        # range_high = mean_x+std_x
        # data_x=rlocdata[:,0]
        # index=np.where([data_x<range_low,data_x>range_high])
        # rlocdata=np.delete(rlocdata,index,axis=0)
        # position_arr=np.mean(rlocdata,axis=0)
        # x=position_arr[0]
```

```
                # y=position_arr[1]
                # z=position_arr[2]




                # TODO: use br.sendTransform (not necessary)



        def get_object_depth(self, obj_struct, cv_image, depth_image, i):

                #
                X_SCALING = 1.333
                Y_SCALING = 0.75
                bbox = obj_struct.bbox
                top_left = (int((bbox.center.x-(bbox.size_x/2))*X_SCALING)), \
                                (int((bbox.center.y-(bbox.size_y/2))*Y_SCALING))
                bottom_right = (int((bbox.center.x+(bbox.size_x/2))*X_SCALING)), \
                                (int((bbox.center.y+(bbox.size_y/2))*Y_SCALING))
        center = (int(bbox.center.x*X_SCALING)),(int(bbox.center.y*Y_SCALING))

        # get average depth
        RECT_ROI_DIST = 15
                #
                if ( center[0]<RECT_ROI_DIST ):
                        center[0] = RECT_ROI_DIST
                elif ( center[0]+RECT_ROI_DIST > IMAGE_WIDTH ):
                        center[0] = IMAGE_WIDTH-RECT_ROI_DIST
                if ( center[1]<RECT_ROI_DIST ):
                        center[1] = RECT_ROI_DIST
                elif ( center[1]+RECT_ROI_DIST > IMAGE_HEIGHT ):
                        center[1] = IMAGE_HEIGHT - RECT_ROI_DIST
                #
                rect_roi_corners = [( center[0]-RECT_ROI_DIST, center[1]-RECT_ROI_DIST ),
\
```

```
                                        (                    center[0]+RECT_ROI_DIST,
center[1]+RECT_ROI_DIST)]
        rect_roi = depth_image[center[0]-RECT_ROI_DIST:center[0]+RECT_ROI_DIST, \
                                                center[1]-
RECT_ROI_DIST:center[1]+RECT_ROI_DIST]
        try:
                avg_depth = int(np.median(rect_roi))
            except:
                avg_depth = np.mean(rect_roi)
                pass

            avg_depth = depth_image[center[1],center[0]]



        if DEBUG_MODE == 1:

                #print('Object: '+str(i)+': '+str(center[0])+','+str(center[1]))

                # write depth of object on images
                cv2.putText(cv_image,          str((avg_depth)),          center,
cv2.FONT_HERSHEY_SIMPLEX, \
                                1.0, (255,255,255), lineType=cv2.LINE_AA)
                cv2.putText(depth_image,          str((avg_depth)),          center,
cv2.FONT_HERSHEY_SIMPLEX, \
                                1.0, (255,255,255), lineType=cv2.LINE_AA)

        # put coords on image
                #cv2.putText(cv_image,          str(top_left),          top_left,
cv2.FONT_HERSHEY_SIMPLEX, \
                # 1.0, (255,255,255), lineType=cv2.LINE_AA)
                #cv2.putText(cv_image,          str(bottom_right),          bottom_right,
cv2.FONT_HERSHEY_SIMPLEX, \
                # 1.0, (255,255,255), lineType=cv2.LINE_AA)
                #cv2.putText(cv_image,          str(bbox),          (100,100),
```

```
cv2.FONT_HERSHEY_SIMPLEX, \
                        # 1.0, (255,255,255), lineType=cv2.LINE_AA)
                #print(str(bbox))

                        # draw bboxes on images
                        cv2.rectangle(cv_image, top_left, bottom_right, (0,0,0), thickness=4)
                        cv2.rectangle(cv_image,  rect_roi_corners[0],  rect_roi_corners[1],  (0,0,0),
thickness=4)
                        cv2.circle(cv_image, center, 2, (255,255,255), -4)
                        cv2.rectangle(depth_image, top_left, bottom_right, (0,0,0), thickness=4)
                        cv2.rectangle(depth_image, rect_roi_corners[0], rect_roi_corners[1], (0,0,0),
thickness=4)
                        cv2.circle(depth_image, center, 2, (255,255,255), -4)

                        # draw line from center of image to center of bbox
                        image_center = (IMAGE_WIDTH/2, IMAGE_HEIGHT/2)
                        cv2.circle(cv_image, image_center, 2, (255,255,255), -4)
                        #
                        cv2.line(cv_image, image_center, center, (255,0,0), 3)
                        # center depth
                        center_depth = depth_image[image_center[1],image_center[0]]
                        cv2.putText(cv_image,          str((center_depth)),          image_center,
cv2.FONT_HERSHEY_SIMPLEX, \
                                        1.0, (255,255,255), lineType=cv2.LINE_AA)

                        signed_height = center[1] - IMAGE_HEIGHT/2
                        signed_width = center[0] - IMAGE_WIDTH/2
                        mid_height = (center[1]+(IMAGE_HEIGHT/2))/2
                        mid_width = (center[0]+(IMAGE_WIDTH/2))/2
                        cv2.line(cv_image, (image_center[0],center[1]), image_center, (255,0,0), 3)
                        cv2.line(cv_image, (image_center[0],center[1]), center, (255,0,0), 3)

                        #cv2.putText(cv_image, 'u='+str((signed_height)), (mid_height,mid_width),
\
```

```
                        #       cv2.FONT_HERSHEY_SIMPLEX,       1.0,        (255,255,255),
lineType=cv2.LINE_AA)
                        #cv2.putText(cv_image, 'v='+str((signed_width)), (mid_height,mid_width),
\
                        #       cv2.FONT_HERSHEY_SIMPLEX,       1.0,        (255,255,255),
lineType=cv2.LINE_AA)


                    #if avg_depth == 0:
                        #print('Object'+str(i)+'detected, but depth reads 0')


            return avg_depth, center, cv_image


        def object_predict(self, object_data, header, image_np,image):
            image_height,image_width,channels = image.shape
            obj=Detection2D()
            obj_hypothesis= ObjectHypothesisWithPose()

            object_id=object_data[0]
            object_score=object_data[1]
            dimensions=object_data[2]

            obj.header=header
            obj_hypothesis.id = object_id
            obj_hypothesis.score = object_score
            obj.results.append(obj_hypothesis)
            obj.bbox.size_y = int((dimensions[2]-dimensions[0])*image_height)
            obj.bbox.size_x = int((dimensions[3]-dimensions[1] )*image_width)
            obj.bbox.center.x = int((dimensions[1] + dimensions [3])*image_height/2)
            obj.bbox.center.y = int((dimensions[0] + dimensions[2])*image_width/2)

            #print(str(obj.bbox))
```

```
# print('height:'+str(image_height))
# print('width: '+str(image_width))

        return obj

def main(args):
    rospy.init_node('detector_node')
    obj=Detector()
    try:
        rospy.spin()
    except KeyboardInterrupt:
        print("ShutDown")
    cv2.destroyAllWindows()

if __name__=='__main__':
    main(sys.argv)
```

Appendix D MoveIt Control Program

```
#include <string>
#include <ros/ros.h>
#include <moveit/move_group_interface/move_group_interface.h>
#include <moveit/planning_scene_interface/planning_scene_interface.h>
#include <moveit_msgs/DisplayRobotState.h>
#include <moveit_msgs/DisplayTrajectory.h>
#include <moveit_msgs/AttachedCollisionObject.h>
#include <moveit_msgs/CollisionObject.h>
#include <vision_msgs/ObjInfo.h>
#include <tf/transform_listener.h>
#include <math.h>
struct Location
{
```

```
    float x;
    float y;
    float z;
};
Location cupCam;
Location cupBase;
int sim=0;
void callback(const vision_msgs::ObjInfo::ConstPtr& revmsg)
{
    cupCam.x=revmsg->x/1000;
    cupCam.y=revmsg->y/1000;
    cupCam.z=revmsg->z/1000;
  // ROS_INFO("recv->obj:%f,%f,%f",cupCam.x,cupCam.y,cupCam.z);
}
void transformPoint(const tf::TransformListener& listener)
{
    geometry_msgs::PointStamped camPoint;
    camPoint.header.frame_id="camera_link";
    camPoint.point.x=cupCam.x;
    camPoint.point.y=cupCam.y;
    camPoint.point.z=cupCam.z;
    try
    {
        geometry_msgs::PointStamped basePoint;
        listener.transformPoint("base_link",camPoint,basePoint);
        if(cupCam.x==0||cupCam.y==0||cupCam.z==0)
            ROS_INFO("Please move leftCup in another position");
        else
            ROS_INFO("LeftCup->cam:%f,%f,%f-
->base:%f,%f,%f",camPoint.point.x,camPoint.point.y\
        ,camPoint.point.z,basePoint.point.x,basePoint.point.y,basePoint.point.z);
          cupBase.x=basePoint.point.x;
          cupBase.y=basePoint.point.y;
          cupBase.z=basePoint.point.z;
```

```
        // ROS_INFO("Transformed location:%f,%f,%f",cupBase.x,cupBase.y,cupBase.z);
    }
    catch(tf::TransformException& ex)
    {
        ROS_ERROR("Fail to find transformation!");
    }
}
void robot_motion(moveit::planning_interface::MoveGroupInterface& arm)
{
    // start of pick
        geometry_msgs::Pose target_pose;
        double raw=180,pitch=0,yaw=0;//in degree
        geometry_msgs::Quaternion q;
        q=tf::createQuaternionMsgFromRollPitchYaw(raw,pitch,yaw);
        target_pose.orientation.x = 0.71;
        target_pose.orientation.y = -0.55;
        target_pose.orientation.z = 0.41;
        target_pose.orientation.w = -0.16;
        // target_pose.orientation.x = 1;
        // target_pose.orientation.y = 0;
        // target_pose.orientation.z = 0;
        // target_pose.orientation.w = 0;

        target_pose.position.x = cupBase.x-0.2;
        target_pose.position.y = cupBase.y;
        target_pose.position.z = cupBase.z+0.2;
        ROS_INFO("Picked obj location:%f,%f,%f",cupBase.x,cupBase.y,cupBase.z);

        if(cupCam.x!=0&&cupCam.y!=0&&cupBase.z>=0.8)
        {

            ROS_INFO("Ready to plan pick.");
            arm.setStartStateToCurrentState();
            arm.setPoseTarget(target_pose);
```

```
            moveit::planning_interface::MoveGroupInterface::Plan plan plan_pick;
            moveit::planning_interface::MoveItErrorCode success = arm.plan(plan_pick);

            ROS_INFO("Plan (pose goal) %s",success?"": "FAILED");

            if(success)
            {
               arm.execute(plan_pick);
               sleep(1);
               ROS_INFO("left waypoints calculated,please run rRightMotion");
               ros::shutdown();
            }

         }

         // }
         // robot_state::RobotState start_state_lift(*arm.getCurrentState());
         //        const        robot_state::JointModelGroup        *joint_model_group        =
start_state_lift.getJointModelGroup(arm.getName());
         // start_state_lift.setFromIK(joint_model_group,target_pose);
         // float GoalZ;
         // GoalZ=cupBase.z+0.2;
         // //directly set waypoints of movel
         // std::vector<geometry_msgs::Pose> waypoints;
         // waypoints.push_back(target_pose);
         // for(;GoalZ>=cupBase.z;GoalZ-=0.02)
         // {
         // target_pose.position.z=GoalZ;
         // waypoints.push_back(target_pose);
         // }
         // moveit_msgs::RobotTrajectory trajectory;
         // const double jump_threshold = 0.0;
         // const double eef_step = 0.01;
         // double fraction = 0.0;
```

```
            // int maxtries = 100; // maximum number of planning attempts
            // int attempts = 0; // number of planning attempts
            // while(fraction < 1.0 && attempts < maxtries)
            // {
            // fraction = arm.computeCartesianPath(waypoints, eef_step, jump_threshold, trajectory);
            // attempts++;
            // //if(attempts % 10 == 0)
            // // ROS_INFO("Still trying after %d attempts..." , attempts);
            // }

            // if(fraction == 1)
            // {
            // ROS_INFO("Path computed successfully. Moving the lArm.");

            // // Generate motion planning data for the robot arm
            // arm.setStartState(start_state_lift);
            // arm.setPoseTarget(target_pose);
            // moveit::planing_interface::MoveGroupInterface::Plan plan_movel;
            // plan_movel.trajectory_ = trajectory;
            // // Execute the campaign
            // arm.execute(plan_movel);
            // ros::shutdown();
            // sleep(1);

            // }
            // else
            // {
            // ROS_INFO("Path planning failed with only %0.6f success after %d attempts.", fraction,
maxtries);
            // }


    }
```

```
    void addCollisionObjects(moveit::planning_interface::PlanningSceneInterface&
planning_scene_interface)
    {
      // BEGIN_SUB_TUTORIAL table1
      //
      // Creating Environment
      // ^^^^^^^^^^^^^^^^^^^^
      // Create vector to hold 3 collision objects.
      std::vector<moveit_msgs::CollisionObject> collision_objects;
      collision_objects.resize(3);

      // Add the first table where the cube will originally be kept.
      collision_objects[0].id = "table_top";
      collision_objects[0].header.frame_id = "base_link";

      /* Define the primitive and its dimensions. */
      collision_objects[0].primitives.resize(1);
      collision_objects[0].primitives[0].type = collision_objects[0].primitives[0].BOX;
      collision_objects[0].primitives[0].dimensions.resize(3);
      collision_objects[0].primitives[0].dimensions[0] = 1.4;//0.74 intotal
      collision_objects[0].primitives[0].dimensions[1] = 1.1;
      collision_objects[0].primitives[0].dimensions[2] = 0.001;

      /* Define the pose of the table. */
      collision_objects[0].primitive_poses.resize(1);
      collision_objects[0].primitive_poses[0].position.x = 0;
      collision_objects[0].primitive_poses[0].position.y = 0.7;
      collision_objects[0].primitive_poses[0].position.z = 0.83;
      // END_SUB_TUTORIAL

      collision_objects[0].operation = collision_objects[0].ADD;

      planning_scene_interface.applyCollisionObjects(collision_objects);
    }
```

```
int main(int argc, char **argv)
{
   ros::init(argc, argv, "LarmMotion");
   ros::NodeHandle n;
   ros::Subscriber sub=n.subscribe("LeftObjects",10,callback);
      tf::TransformListener listener(ros::Duration(1));
      ros::Timer
timer=n.createTimer(ros::Duration(1),boost::bind(&transformPoint,boost::ref(listener)));


      while(ros::ok)
        {
          ros::spinOnce();
          ros::AsyncSpinner spinner(1);
          spinner.start();

          moveit::planning_interface::MoveGroupInterface arm("left_manipulator");


          //Get the name of the terminal link
          std::string end_effector_link = arm.getEndEffectorLink();

          // Set the reference coordinate system to be used for the target position
          std::string reference_frame = "base_link";
          arm.setPoseReferenceFrame(reference_frame);

          // Allow re-planning when motion planning fails
          arm.allowReplanning(true);

          //Setting the allowable error of position (in meters) and attitude (in radians)
          arm.setGoalPositionTolerance(0.001);
          arm.setGoalOrientationTolerance(0.01);
```

```
            //Setting the maximum speed and acceleration allowed
            arm.setMaxAccelerationScalingFactor(0.2);
            arm.setMaxVelocityScalingFactor(0.2);
            //add object

            // Define a pose for the box (specified relative to frame_id)
            // Define a collision object ROS message.
            moveit::planning_interface::PlanningSceneInterface planning_scene_interface;
            addCollisionObjects(planning_scene_interface);

            robot_motion(arm);

        }



    return 0;
}
```

Appendix E Action Client Program

```
#include <ros/ros.h>
#include <actionlib/server/simple_action_server.h>
#include <control_msgs/FollowJointTrajectoryAction.h>
#include <trajectory_msgs/JointTrajectory.h>
//header for socket

#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
```

Appendix

```
#include <string.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/shm.h>
#include <vision_msgs/ObjInfo.h>
#include <tf/transform_listener.h>
#define PI 3.1415
#define PORT 7000
#define BUFFER_SIZE 1024
char endbuf[600];
char sendbuf[55];
char recvbuf[BUFFER_SIZE];
char tmp_cmd[BUFFER_SIZE];
int sock_cli = socket(AF_INET, SOCK_STREAM, 0);
using namespace std;
struct Location
{
   float x;
   float y;
   float z;
};
Location cupCam;
Location cupBase,lcupBase;
struct Pose
{
 float x;
 float y;
 float z;
 float rx;
 float ry;
 float rz;
};
Pose leftCup;
//socket client ,action server
```

```
class JointTrajectoryActionServer
{
public:
JointTrajectoryActionServer(std::string name):
        as_(nh_, name, false), action_name_(name)
{
 // register callback for goal
 as_.registerGoalCallback(boost::bind(&JointTrajectoryActionServer::goalCallback, this));
 as_.start();
}
~JointTrajectoryActionServer(void){}


// when a trajectory command comes, this function will be called.
void goalCallback()
{
 boost::shared_ptr<const control_msgs::FollowJointTrajectoryGoal> goal;
 goal=as_.acceptNewGoal();
 int point_size=goal->trajectory.points.size();
 cout<<"trajectory point size:"<< goal->trajectory.points.size()<<endl;
 int num=0;
 int joint_num=0;
 float wayPoints;
 int counter=0;
 int wayPointsNum=0;
 wayPointsNum=goal->trajectory.points.size();
 //store wayPoints
 float PointStor[goal->trajectory.points.size()][7];

 for(;num<goal->trajectory.points.size()-1;num=num+5)
 {
        if(num==0)
        {
                strcpy(sendbuf,"/RP/ ");
```

```
        send(sock_cli,sendbuf,sizeof(sendbuf),0);
        cout<<sendbuf<<endl;
}
float GoalDeg[7]={ goal->trajectory.points[num].positions[0]*(180/PI),
                                        -goal->trajectory.points[num].positions[1]*(180/PI),
                                        goal->trajectory.points[num].positions[2]*(180/PI),
                                        goal->trajectory.points[num].positions[3]*(180/PI)+90,
                -goal->trajectory.points[num].positions[4]*(180/PI),
                                         -goal->trajectory.points[num].positions[5]*(180/PI),
                                       goal->trajectory.points[num].positions[6]*(180/PI)};

// cout<<"point_number"<<num+1< ":[" << (GoalDeg[0])
      // << "," << (GoalDeg[0])
      // << "," << (GoalDeg[1])
      // << "," << (GoalDeg[2])
      // << "," << (GoalDeg[3])
      // << "," << (GoalDeg[4])
// << "      ," << (GoalDeg[5])
// <<        "," << (GoalDeg[6])
      // << "]" << endl ;

int i;
for(i=0;i<7;i++)
{
      PointStor[num][i]=GoalDeg[i];
}
//debug information
  sprintf(sendbuf, "movej([%.0f,%.0f,%.0f,%.0f,%.0f,%.0f,%.0f,%.0f],30,2000,-1); ",
      GoalDeg[0],
              GoalDeg [1],
              GoalDeg [2],
              GoalDeg [3],
            GoalDeg [4],
             GoalDeg [5],
```

```
                GoalDeg [6]);
                //send the cmd to the robot
            send(sock_cli,sendbuf,sizeof(sendbuf),0);
            cout<<sendbuf<<endl;


        }
            float endGoal[7]={goal->trajectory.points[point_size-1].positions[0]*(180/PI),
                                                -goal->trajectory.points[points_size-
1].positions[1]*(180/PI),
                                               goal->trajectory.points[point_size-1].positions[2]*(180/PI),
                                                goal->trajectory.points[point_size-
1].positions[3]*(180/PI)+90,
                            -goal->trajectory.points[point_size-1].positions[4]*(180/PI),
                                        -goal->trajectory.points[point_size-1].positions[5]*(180/PI),
                                      goal->trajectory.points[point_size-1].positions[6]*(180/PI)};



        // if (wayPointsNum%2==0)
        // {


        // }
                sprintf(endbuf,         "movej([%.0f,%.0f,%.0f,%.0f,%.0f,%.0f,%.0f,%.0f,%.0f],30,2000,-
1);movej_pose([%.0f,%.0f,%.0f,90,40,-95],40,20,-1);movej_pose(       [%.0f,%.0f,%.0f,%.0f,90,40,-
95],40,20,-1);sleep(6);<end>",
                    endGoal[0],
                            endGoal[1],
                            endGoal [2],
                            endGoal [3],
                       endGoal [4],
                          endGoal [5],
                    endGoal [6],
                    cupBase.x*1000+400,//1
                    cupBase.y*1000-20,
                    cupBase.z*1000+250,
```

```
                cupBase.x*1000+270,//2
                cupBase.y*1000-30,
                cupBase.z*1000+10);
            //send the cmd to the robot
            send(sock_cli,endbuf,sizeof(endbuf),0);
            cout<<endbuf<<endl;

        /**sprintf(sendbuf,                    "movel([%f,%f,%f,%f,%f,%f,%f,%f],30,2000,-
1);movep([%f,%f,%f,%f,%f,%f+90,%f],30,2000,-1);\
                    movep([%f,%f,%f,%f,%f,%f,%f],30,2000,-
1);",rightCup.x,rightCup.y,rightCup.z,rightCup.rx\
                    rightCup.ry,rightCup.rz,rightCup.x,rightCup.y,rightCup.z,rightCup.rx\
                    rightCup.ry,rightCup.rz,rightCup.x,rightCup.y,rightCup.z,rightCup.rx\
                    ,rightCup.ry,rightCup.rz);**/
        //send(sock_cli,sendbuf,sizeof(sendbuf),0);
        // tell motion control hardware to execute
            // do something
        // when finished, return result
        as_.setSucceeded(result_);
      }
      protected:
      ros::NodeHandle nh_;
      actionlib::SimpleActionServer<control_msgs::FollowJointTrajectoryAction> as_;
      actionlib::SimpleActionServer<control_msgs::FollowJointTrajectoryAction>::Result result_;
      std::string action_name_;
      };

      void init_socket()
      {
      //define sockfd

            //define sockaddr_in
            struct sockaddr_in servaddr;
            servaddr.sin_family = AF_INET;
```

```
        servaddr.sin_port = htons(PORT); //server port
        servaddr.sin_addr.s_addr = inet_addr("127.0.0.1"); //use local addr to test
        if(connect(sock_cli, (struct sockaddr *)&servaddr, sizeof(servaddr)) < 0)
        {
                perror("connect");
                exit(1);
        }
}

void callback(const vision_msgs::ObjInfo::ConstPtr& revmsg)
{
   cupBase.x=revmsg->x;
   cupBase.y=revmsg->y;
   cupBase.z=revmsg->z;
}
void lcallback(const vision_msgs::ObjInfo::ConstPtr& revmsg)
{
   lcupBase.x=revmsg->x;
   lcupBase.y=revmsg->y;
   lcupBase.z=revmsg->z;
}
int main(int argc, char** argv)
{
 ros::init(argc,argv, "Rserver");
 ros::NodeHandle n;

 init_socket();
 //JointTrajectoryActionServer srv("left_manipulator_controller/follow_joint_trajectory");
 JointTrajectoryActionServer srv("right_manipulator_controller/follow_joint_trajectory");
 ros::Subscriber rsub=n.subscribe("RightObjBase",10,callback);
 ros::Subscriber lsub=n.subscribe("LeftObjBase",10,lcallback);
 ros::spin();
 return 0;
}
```

Appendix