“Vision-Based 6DOF Manipulator Grasping System”

By

Ziyu Du

H00263856

An Honours report submitted for the Degree of:

BEng in Electrical and Electronic Engineering

Supervised by

Sen Wang

April 2020

Synopsis

This thesis is about how a vision-based 6-DOF manipulator Grasping System is built, including establishment of simulation environment, camera calibration, object recognition schemes, pose estimation and grasping. Besides, two methods of recognition will be compared and analyzed in this thesis. Finally, at the end of thesis, I will analyze whether the results of project obtained are acceptable and some difficulties and shortcoming will be pointed out.

Acknowledgement

I would like to extend my heartfelt thanks to my supervisor Sen Wang, for his instructive advice and suggestions on my project. I am deeply grateful of his help in the completion of this thesis.

I am also deeply indebted to tutor Cong Wang for guiding my project and pointing out problems.

Finally, I am indebted to my teammates, Yuanyue You, Zhanheng Li, Kaicheng Zhang and Zhuocheng Han for useful team discussion.

Statement of Authorship

I, Ziyu Du

State that this work submitted for assessment is my own and expressed in my own words. Any uses made within it of works of other authors in any form (eg. Ideas, figures, text, tables) are properly acknowledged at their point of use. A list of the references employed is included.

Date.....21/04/2020.....

Index

Figure and table list

1. Introduction

1.1 the objective

Human hands are the most efficient tools to achieve manipulation and classification, especially combined with human visual recognition. But, as the industrial scale expands and industrial processes become more complex, industry needs to introduce robotic arms to replace human hands. In my project, I intend to design a robotic arm system based on robot vision and robot operating system. The objective of my project is to control a manipulator by visual information feedback. The camera hopefully would be able to recognize target and publish its reliable depth information to computer.

1.2 Overview

The construction of the system requires the support of various equipment. In details, I choose universal robot which attached a 2-finger-85cm-Robotiq gripper as manipulation part (important for project integrity), besides, as for vision recognition part, Kinect series released by Microsoft in 2010 is perfect for my project according to its excellent compatibility with ROS and RGBD camera features. It is worth mentioning that I will build a reliable manipulation system in Gazebo, a physical simulation platform.

1.3 The Structure of Thesis

There are four main parts in this thesis, the first part introduces project and overview briefly.

In the second part, it mainly describes the background knowledge used in project, including the communication method used in ROS, basic camera calibration knowledge, matrix conversion from pixel coordinate to world coordinate, basic knowledge about scale-invariance feature transform method, Actionlib and Moveit introduction, inverse kinematics solver, motion planning introduction and project equipment parameters.

In the third part, it mainly describes the establishment procedure of vision-based manipulator grasping, including working environment introduction, target recognition

scheme, comparison SIFT with OpenCV method and manipulator grasping results. Besides, there are some problems discussed in third part.

2. Background

2.1 Robot operating System

What is ROS [1]，“ The robot operating system is a set of open source software libraries and tools that help you build robot application ”, from ROS official homepage. Nowadays, ROS platform is widely used in mobile robot, aircraft, binocular camera, robot arm and other fields. The purpose of ROS is to build a general robot original platform integrating different research results, realizing algorithm release and code reuse. ROS has the following main features: distributed architecture, various language support, good scalability, open sources.

The developing of robot arm is inseparable from a good coding environment. According to these advantages, I select ROS as a software platform of manipulator grabbing. Meanwhile, ROS has good compatibility with OpenCV library [2] of machine visions which is convenient to process the collected image, so as to realize the object recognition and pose estimation. A visualization platform RVIZ is also including in ROS platform which can display three-dimensional information, intuitively showing that the geographical relationship among the manipulator, camera and target object.

2.1.1 Communication Modes

now I can learn that ROS provides several communication modes for robot and large number of packages which help robot achieve various functions. It is very important to understand the communication method in ROS, therefore I will introduce three communication methods I used in my project: topic and service and action communication modes. Before talking about the three communication methods, first importantly, the data structure should be introduced. Messages play important role in the whole ROS communication system. Messages [3] are simple data structure containing the information which will be transmitted in channel. There are a large number of message types. For example, the messages type in /KinectV2/ rgb/ Image_raw topic is sensor_msgs/ Image.

As for topic [4] method, it is a named bus for ROS node to exchange messages. Besides, topic can subscribe message to master and publish messages from master anonymously, which means the generation and consumption of messages is decoupled. There is no forced binding relationship between this two. It only cares about the topic name and current message type when it matches my requirement. The following picture vividly explain the communication mechanism of ROS. The Publisher publish a topic called /topic and Subscriber subscribe this topic, meanwhile Subscriber can get the message data through /topic channel from Publisher.

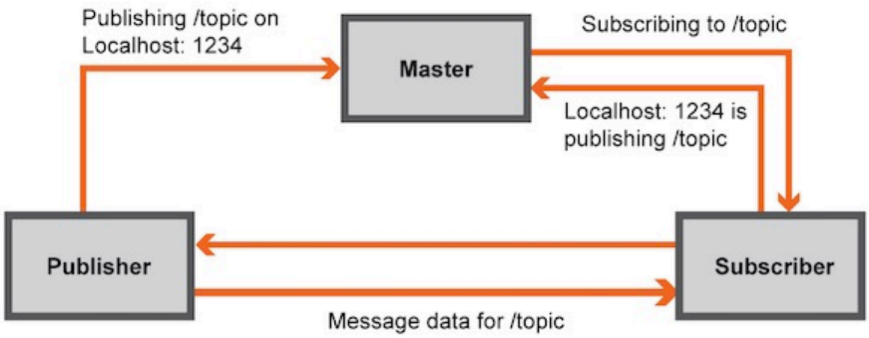


Figure 1.ROS communication modes

ROS service [5] is an interactive form based on request and reply form. It is mainly used distributed system. ROS service is defined by pair of messages, one of which is used to define the request data type and the other is used to define the reply data type. In ROS service, a node acts as a ROS service. A client can request a service from this service, and after the service, it will send the result to its client.

Action method can be considered as advanced service communication method, but action contains five topics, goal, status, cancel, feedback, result, while service only contains two topics, request and reply. Action service is quite useful in real-time process.

2.2 Camera Calibration Method

The basic idea of camera calibration is to solve the problem in determining the correlation between three-dimensional geometric position of a point on a surface of a space object and its corresponding point in the image. Therefore, the result of camera calibration is to get the camera parameters.

In my project, I decided to use Zhang camera calibration [6] which is a method between photogrammetric calibration and self-calibration according to its flexibility with one checkboard and reliable precision.

Camera Model

The Zhang camera model [7] is setting the 2D point denoted by m = [u v 1]. A 3D point in world coordinate is denoted by M = [X Y Z 1], therefore the relationship between. checkboard plane used for calibration to the image plane is that:

$$sm = A[R|\ t]M \tag{1}$$

where ***s*** is an arbitrary scale factor while [R [8] | t] is camera extrinsic matrix, R is called rotation matrix which is a 3x3 matrix calculated by Euler angle: pitch angle p, yaw angle y and roll angle r

$$R = \begin{bmatrix} cosr & sinr & 0 \\ -sinr & cosr & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cosy & siny \\ 0 & -siny & cosy \end{bmatrix} \begin{bmatrix} cosp & sinp & 0 \\ -sinp & cosp & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

In the （1） function parameter t is a 3x1 matrix called translation matrix and t contains the information that relative position in the world coordinate. The most important parameter in function is A, the camera intrinsic matrix. The aim of calibration process is to get the camera intrinsic matrix A [7].

$$A = \begin{bmatrix} a & G & u0 \\ 0 & b & v0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Where a and b are scale factor in image u and v axis, G is the degree of skewness between axis u and v. Then assume that model plane is Z = 0 and divide R into r, therefore from (1) we can get:

$$\text{sm} = \text{A}[\text{r1 r2 t}] \begin{bmatrix} \text{X} \\ \text{Y} \\ 1 \end{bmatrix} \tag{4}$$

Function (4) means making M denote a point on checkboard where Z = 0, hence, is it convenient to mix camera matrix get H [9]:

$$\mathrm{sm} = \mathrm{HM} \tag{5}$$

$$\mathrm{H} = \mathrm{A}[\mathrm{r1\ r2\ t}] = [\mathrm{h1\ h2\ h3}] \tag{6}$$

As we can see (6), H parameter is a 3x3 matrix where h1, h2 and h3 are 3x1 matrix respectively. The total description for the Zhang camera model has been completed, then the calculation part should be discussed.

Constraints [10]

Before solving the camera intrinsic matrix, some constraints of the intrinsic matrix should be discussed.

The first constrain is that the dot product of rotation vector is 0. It is because the rotation vectors in two vertical planes are perpendicular to each other. The formula derivation is:

$$\mathrm{r1}^{\mathrm{T}}\mathrm{r2} = 0 \text{ while } \mathrm{r1} = \lambda \mathrm{A}^{-1}\mathrm{h1};\ \ \mathrm{r2} = \lambda \mathrm{A}^{-1}\mathrm{h2}\ ;\ \lambda = \frac{1}{\mathrm{s}} \tag{7}$$

Therefore, from the first constraints I can get:

$$\mathrm{h1}^{\mathrm{T}}(\mathrm{A}^{-1})^{\mathrm{T}}\mathrm{A}^{-1}\mathrm{h2} = 0 \tag{8}$$

Rotation vectors are equal length because rotation does not change scale, therefore the equation can be obtained that:

$$\mathrm{h1}(\mathrm{A}^{-1})^{\mathrm{T}}\mathrm{A}^{-1}\mathrm{h1} = \mathrm{h2}(\mathrm{A}^{-1})^{\mathrm{T}}\mathrm{A}^{-1}\mathrm{h2} \tag{9}$$

Solving Camera Matrix [11]

Two constraints [8] and [9] are prepared for calculating the camera matrix, then A is the camera matrix, therefore $(A^{-1})^T A^{-1} = B$*(10)* should be calculated.

$$(A^{-1})^T A^{-1} = \begin{bmatrix} \frac{1}{a^2} & -\frac{G}{a^2 b} & \frac{v0G-u0b}{a^2 b} \\ -\frac{G}{a^2 b} & \frac{G^2}{a^2 b^2}+\frac{1}{b^2} & -G\frac{v0G-u0b}{a^2 b^2}-\frac{v0}{b^2} \\ \frac{v0G-u0b}{a^2 b} & -G\frac{v0G-u0b}{a^2 b^2}-\frac{v0}{b^2} & \frac{(v0G-u0b)^2}{a^2 b^2}+\frac{v0^2}{b^2}+1 \end{bmatrix} = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix}$$

As the calculation process shows the matrix B is a symmetric matrix, therefore there are only 6 useful matrix elements which are on either side of the main diagonal.

Then take B into (8) and (9), the two constraints become:

$$\mathrm{h1^{T}Bh2} = 0 \tag{11}$$

$$\mathrm{h1Bh1} = \mathrm{h2Bh2} \tag{12}$$

That two equations can be written into general equation:

$$h_i{}^T B h_j \tag{13}$$

From function [6], the 3x3 matrix for the ith column vector can be defined that:

$$h_i = [\mathrm{h_{i1}\ h_{i2}\ h_{i3}}]^{\mathrm{T}} \tag{14}$$

As we can see, it is important to calculate around two constraints during calculation.

Then take the formula [14] into [13]

$$h_i{}^T B h_j = \begin{bmatrix} h_{i1}h_{j1} \\ h_{i1}h_{j2} + h_{i2}h_{j1} \\ h_{i2}h_{j2} \\ h_{i3}h_{j1} + h_{i1}h_{j3} \\ h_{i3}h_{j2} + h_{i2}h_{j3} \\ h_{i3}h_{j3} \end{bmatrix} \begin{bmatrix} B_{11} \\ B_{12} \\ B_{22} \\ B_{13} \\ B_{23} \\ B_{33} \end{bmatrix} \tag{15}$$

While $v_{ij=} \begin{bmatrix} h_{i1}h_{j1} \\ h_{i1}h_{j2} + h_{i2}h_{j1} \\ h_{i2}h_{j2} \\ h_{i3}h_{j1} + h_{i1}h_{j3} \\ h_{i3}h_{j2} + h_{i2}h_{j3} \\ h_{i3}h_{j3} \end{bmatrix}, b = \begin{bmatrix} B_{11} \\ B_{12} \\ B_{22} \\ B_{13} \\ B_{23} \\ B_{33} \end{bmatrix}$

Finally, the constraints can be transformed into:

$$v_{12}{}^T b = 0 \tag{16}$$

$$(v_{11}{}^T - v_{22}{}^T)b = 0 \tag{17}$$

If we take n number of calibration pictures of different angles, we can get a set of the above equations [12]. Among them, v11, v12 and v22 can be obtained through the H that has been calculated previously. However, the six elements in b matrix is unknown to be found. Therefore, the number of pictures is at least n=3. Then the elements in intrinsic matrix A can be calculated:

$$a = \sqrt{\lambda / B_{11}}$$

$$b = \sqrt{\lambda B_{11}/(B_{11}B_{22} - B_{12}^2)}$$

$$u0 = \frac{Gv0}{b} - \frac{B_{13}b^2}{\lambda}$$

$$v0 = (B_{12}B_{13} - B_{11}B_{23})/(B_{11}B_{22} - {B_{12}}^2)$$

$$G = -B_{12}a^2b/\lambda$$

$$\lambda = B_{33} - [B_{13}^2 + v0(B_{12}B_{13} - B_{11}B_{23})]/B_{11}$$

Therefore, we can calculate the matrix A by taking n>3 number calibration pictures of different angles to get B.

2.3 Object Recognition and Coordinate Conversion

Object recognition is an important part of the whole project, object recognition connects the robot arm with the world, which means I can get identify the target figure object from other interferences and positioning the target object in a world coordinate centered on the base of robot arm. Between the identifying and positioning process, there is a process called coordinate matrix conversion. As a result of the location information gotten from object recognition is pixel information. Therefore, the matrix conversion is necessary.

2.3.1 SIFT

The full name of sift is scale invariant feature transform, which was proposed by Canadian professor David g. Lowe. SIFT [13] feature is a very stable local feature, SIFT is used in my project according to its invariant to rotation, scaling, brightness change and convenient application in find-object function package. Here are four main steps in invariant feature transform and its flow chart is as followed:

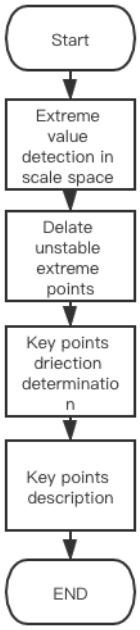


Figure 2. SIFT algorithm steps flow chart

Gaussian Blur [14]

Scale-invariant feature transform method is a computer vision algorithm used to detect and describe feature in images. The key of the algorithm published by David Lowe in 1999, perfected in 2004, is to find the maximum and minimum of the difference of Gaussian function applied in scale space and extracts their position, scale and rotation invariants information.

The two dimensional Gaussian low-pass filter play an important role in scale-invariant feature transform method. The two dimensional Gaussian low-pass filter [15] is a kind of digital filter which can do convolution operation with the original picture to achieve the purpose of blurring the image.

The N-Dimensional Gaussian distribution function [16]:

$$G(r) = \frac{1}{\sqrt{2\pi\sigma^2}^N} e^{-r^2/(2\sigma^2)}$$

Where σ is the standard deviation of the normal distribution and the value of σ control the degree of image ambiguity, the larger σ is, the more blurred the image is. Where r

is called Blur radius which refers to distance from the template elements to the center of the template. Assuming that the element in template is(x,y) and the scale of two dimensional template is m*n, therefore r is (x-m/2)^2+(y-n/2)^2. The N-dimensional Gaussian distribution function becomes that:

$$G(x,y)=\frac{1}{2\pi\sigma^2}e^{-\frac{(x-m/2)^2+(y-n/2)^2}{2\sigma^2}}$$

In two-dimensional space, the aim of this function is to generate concentric circle with center starting to appear normally distributed. As we can see in figure 2, the value of each original pixel has the largest Gaussian distribution, therefore it has the largest weight; the neighboring pixels get farther and farther away from the original pixel, their weight gets smaller and smaller. Then, after convolving each pixel of the original image, we can get a total average weighted new image.

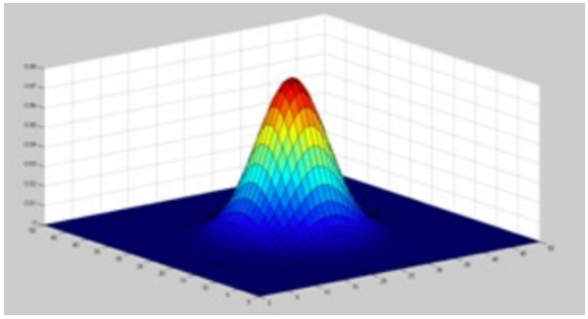


Figure 3. Two-Dimension Gaussian surface

Check the 1-Dimension Gaussian distribution again. σ, as I said before, is an important parameter that can change the maximum weight of the Gaussian curve. As we can see in figure 3, the larger σ, the smaller the weight is. It is the same to the 2-Dimension Gaussian distribution.

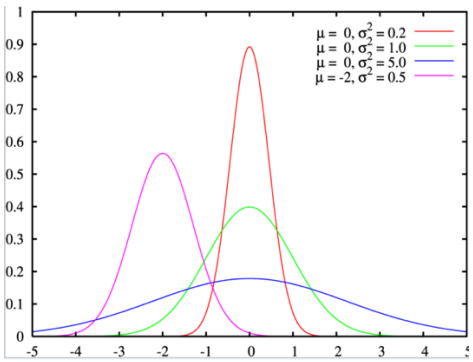


Figure 4. One-Dimension Gaussian with changing σ

However, the use of a two-dimension Gaussian template blurring the image will cause the edge image which means that there are black edges at the edges of the image. In order to solve this problem, R. Gonzalez and R. Woods told that we can use two one-dimension Gaussian to achieve removing black edges [19].

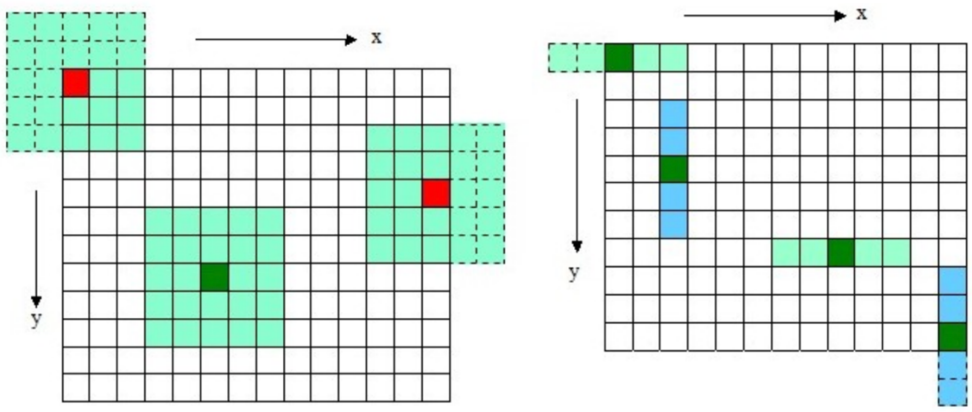


Figure 5. transform from 2-dimension Gaussian blur to 1-dimension Gaussian blur

Gaussian blur calculation run through the entire SIFT algorithm, there are four steps about SIFT algorithm written by Lowe, extreme value detection in scale space, delating unstable extreme points, key points direction determination and key points description.

Extreme Value Detection in Scale Space

First of all, what is scale space? According to *Tony Lindeberg, scale-space theory: A basic tool for analyzing at different scales*. Scale space [18] is a framework for controlling the observation scale or characterizing the multi-scale natural characteristics of image data. In SIFT algorithm, the Gaussian pyramid model can well explain the algorithm process.

The purpose of constructing scale space is to find positions that are invariant in scale changes. Continuous scale changes can be used, that is, to find stable feature points in all possible scale changes in scale space. The poles found in this way can ensure that Invariant in image scaling and rotation changes.

Assuming I (x, y) is the original image, G (x, y, σ) s a Gaussian function with variable scale space, then the scale space [19] of an image can be defined as:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y). \quad (18)$$

Where $*$ is the convolution operation, σ indicates the size of the scale space, and the larger σ indicates the higher blur.

While in Lowe's paper, he rewrites the Gaussian function [19]:

$$G(x, y, \sigma) = \frac{1}{2pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \quad (19)$$

In Lowe's paper, the process of generating large scale space image L (x, y, σ2) from a smaller scale space image L (x, y, σ1) is:

$$L(x, y, \sigma 2) = G\left(x, y, \sqrt{\sigma 2^2 - \sigma 1^2}\right) * L(x, y, \sigma 1) \quad (20)$$

Where

$$G\left(x, y, \sqrt{\sigma 2^2 - \sigma 1^2}\right) = \frac{1}{2pi\sqrt{\sigma 2^2 - \sigma 1^2}^2} e^{-(x^2+y^2)/2\sqrt{\sigma 2^2 - \sigma 1^2}^2} \quad (21)$$

Therefore, in order to find the stable maximum and minimum point in scale space. Gaussian difference (DOG) function is used in SITF algorithm.

$$D(x, y, \sigma) = [G(x, y, k\sigma) - G(x, y, \sigma)] * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma) \quad (22)$$

In the function (22), the smooth factors are kσ and σ, where the smooth factor in every octave in Gaussian pyramid is 0, σ, kσ, k^2σ, k^3σ…… We can consider that the DOG image is the difference between the two adjacent images with the same scale and different smooth factors, therefore the figure structure [20] is:

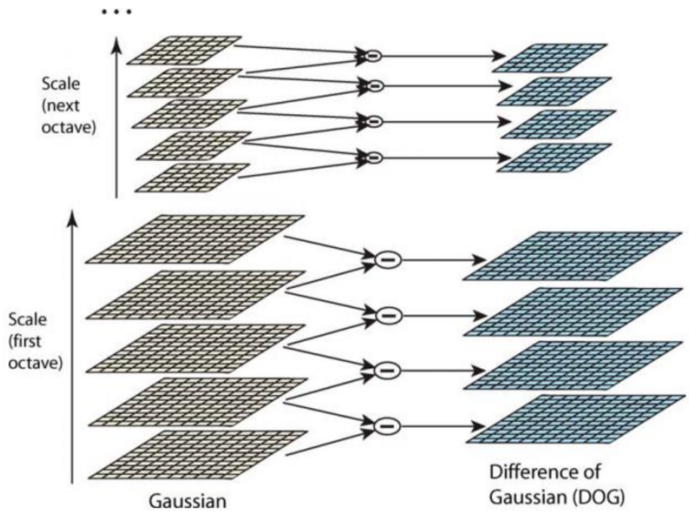


Figure 6. The generation of DOG plane

As Lowe's paper said before, the aim of DOG is to find the extreme point which represents some stable feature points in the scale space. In order to find the maximum

and minimum points, every pixel point should be compared with the neighboring points to check whether it is larger or smaller than the adjacent points.

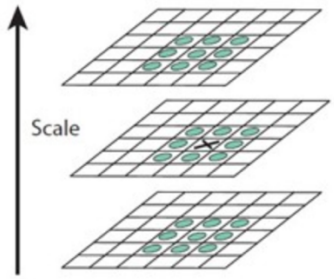


Figure 7. Extreme point detection in DOG plane

In figure 6, the process describes assumed maximum or minimum point should be compared with the 26 neighboring points [20] to get the extreme key points. If it is a maximum or minimum pixel point at this level, then the closest pixel location is calculated at the next level of pyramid. However, the extreme points detected in discrete space is not real extreme point. Therefore, the interpolation method should be introduced into SIFT algorithm to optimize the choices of extreme points and improve the stability of key points.

Delate Unstable Extreme Points

The second step of SIFT algorithm is to delate unstable extreme points gotten from the DOG detection steps. The local extreme points of DOG obtained by comparison detection are obtained by discrete space search. Because discrete space is the result of continuous space sampling, the extreme points found in discrete space are not necessarily the real extremum points, so Lowe told that we should try to eliminate the points that do not meet the conditions. We can find the extremum point by curve fitting with the scale space DOG function. The essence of this step is to remove the points with very asymmetric local curvature of the DOG. The two kinds of key points that do not meet the requirement is that:

1. Low contrast key points [21]
2. Unstable edge response points [22]

Filtering the low contrast key points means the extreme points gotten from the detection step are not necessarily the real extreme points. Therefore, delating the low

contrast key points can make key points more accurate. In Lowe's paper, Taylor expansion equation can be applied into optimization key points and a threshold value will be set. The main solution idea is that making Taylor expansion of function [22] and calculating the extreme value by doing differentiation of function [22] and comparing this calculated value with the threshold value. The absolute value of any solution is considered as feasible as long as it is greater than the threshold value.

Unstable edge response points are generated by the Gaussian smoothing filtering of the original image which cause the main curvature value is large in the direction of edge gradient, while the main curvature value is small along the edge direction. In Lowe's paper, principal curvature of the D (x) of the candidate key points is directly proportional to the eigenvalue of the 2 × 2 hessian matrix H which describes local curvature value of function. The main idea is that calculating the trace and determinant of hessian matrix H and comparing trace and determinant values with the given threshold value. Therefore, the key points that satisfy the following inequality can be kept, otherwise, it will be rejected.

$$\frac{(Tr(H))^2}{Det(H)} > \frac{(T_l+1)^2}{T_l} \tag{23}$$

Where Tr(H) is the trace value of Hessian matrix of candidate key points and Det(H) is the determinant value of Hessian matrix of candidate key points, T_l is the threshold value which describe the ratio of the maximum eigenvalue to the minimum eigenvalue, in Lowe's paper, T_l is set to 10.

Key Points Direction Determination [23]

The aim of SIFT algorithm is to recognize the target and get its pixel coordinate information. After above steps, we can find key points in scale space. In order to achieve image rotation invariance, a fine fitting model to determine the location and scale should be made. The selection of key points depends on their stability. Then one or more directions of each key position are assigned based on the local gradient direction of the image. All subsequent operations on image data are transformed with

respect to the direction, scale and position of key points, so as to provide invariance for these transformations.

For the key points detected in the dog pyramid, the gradient and direction distribution characteristics of the pixels in the 3 σ neighborhood window of the Gaussian pyramid image are collected:

$$\mathrm{Mij} = \sqrt[2]{(\mathrm{A_{ij}} - \mathrm{A_{i+1j}})^2 + (\mathrm{A_{ij}} - \mathrm{A_{ij+1}})^2} \tag{23}$$

$$\mathrm{Rij} = \operatorname{atan}\left(\left(\mathrm{A_{ij}} - \mathrm{A_{i+1j}}\right)^2 + \left(\mathrm{A_{ij}} - \mathrm{A_{ij+1}}\right)^2\right) \tag{24}$$

Where Mij is the image gradients magnitude and Rij is image gradient orientation and Aij is each pixel.

After calculating the gradient direction, we need to use histogram to count the gradient direction and amplitude of the pixels in the neighborhood of the feature points. The horizontal axis of histogram of gradient direction is the angle of gradient direction. The vertical axis is the accumulation of the gradient amplitude corresponding to the gradient direction, and the peak value of the histogram is the main direction of the key point. In Lowe's paper, we use Gaussian function to smooth histogram to enhance the effect of neighborhood points near feature points on the direction of key points and reduce the impact of mutation. In order to get more accurate direction, the discrete gradient histogram can also be interpolated. Specifically, the direction of the key point can be obtained by parabola interpolation of the three column values closest to the main peak value. In the gradient histogram, when there is a column value equivalent to 80% of the energy of the main peak value, this direction can be considered as the auxiliary direction of the key point. Therefore, a key point may detect multiple directions (it can also be understood that a feature point may generate multiple feature points with the same coordinates and scales, but different directions).

15% of key points are multidirectional. After getting the main direction of the key points, three information (x, y, σ, θ) can be obtained for each key point, location, scale and direction. Therefore, a SIFT feature area can be determined by three values,

the center represents the location of key points, the radius represents the scale of key points and the arrow represents the main direction of key points. Key points with multiple directions can be copied into multiple copies, and then the direction values are assigned to the copied feature points respectively. multiple key points with equal coordinates and scales, but different directions can be generated by a key point

Key Points Description [24]

The final step is to describe the key point into a vector format which is keeping unchanged with various changes such as changes in light, angles of views. This descriptor includes not only the key points, but also the pixels around the key points that contribute to it. Moreover, the descriptor should have high uniqueness, so as to improve the probability of correct matching of feature points.

Firstly, in order to ensure the rotation invariance of the key point vector, we should take the key point as the center, and rotate the coordinate axis θ (the main direction of the feature point) angle in the neighborhood, that is, the coordinate axis is rotated to the main direction of the key point.

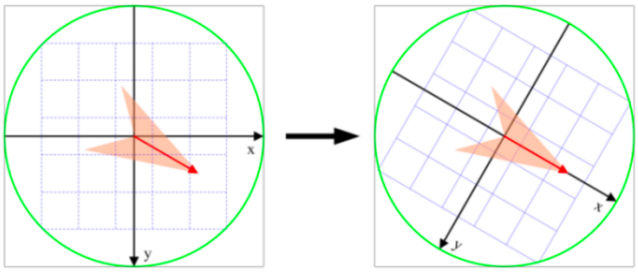


Figure 8. Correct the main direction of rotation

After rotating process, getting 8x8 window centered on main direction, each small lattice represents a pixel in the scale space of the key point neighborhood. The gradient amplitude and gradient direction of each pixel are calculated. The arrow direction represents the gradient direction of the pixel, and the length represents the gradient amplitude. Then, the Gaussian window is used to weight it. At last, eight directions of gradient histogram are drawn on every 4 × 4 small block, and the accumulated value of each gradient direction is calculated to form a seed point.

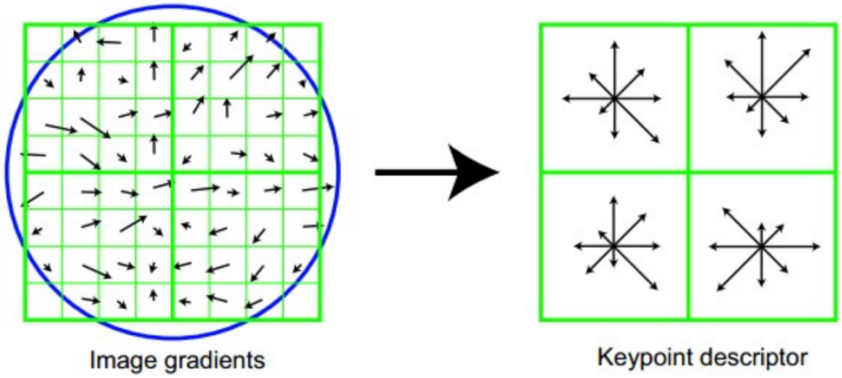


Figure 9. 2x2 key point descriptor

In the actual calculation process, in order to enhance the robustness of matching, Lowe suggests using 4 × 4 16 seed points to describe each key point, so that a key point can generate 128 dimensional SIFT feature vector.

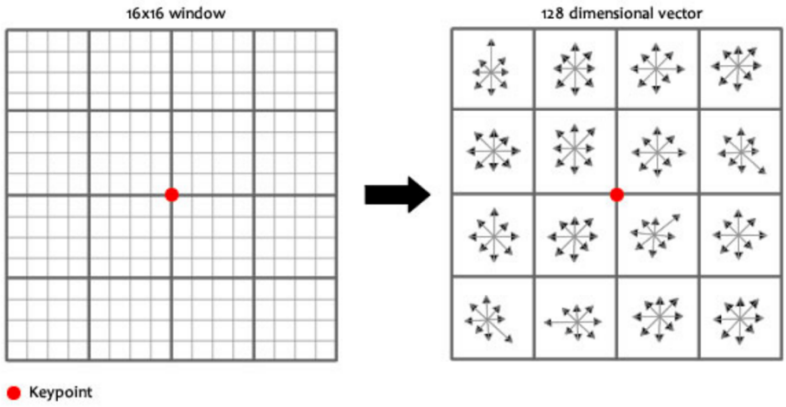


Figure 10. 4x4 128-dimensional key point descriptor.

2.4 Poses Estimation

From previous chapter, I can get a two-dimensional pixel coordinate point which represents the reconfiguration coordinate point of several feature points from SIFT [13] algorithm. In order to get three-dimensional corresponding world coordinates, the manipulator-camera model and coordinates conversion are necessary.

In general, there are two ways of machine grasping system, eye-to-hand [25] and eye-in-hand [25]. In the eye-in-hand mode, the camera is attached to the end-effect of robot arm, in the process of grasping, camera moves with the movement of the robot arm. However, in the eye-in-hand mode, there are some disadvantages should not be

ignored. When the end-effect is closed to target item, it will cause the vision losses of the target object and it is not easy to design control system. Therefore, in my project, the eye-to-hand mode is selected owing to ease access to the coordinate relationship between the target object and camera.

In this part, the main problem is to study the grasping problem of the known object under the position and angle of the manipulator base coordinate. The main contents include position and angle matrix conversion, introduction of ROS Moveit, inverse kinematics solver and motion planner.

2.4.1 Target Pose Relation Conversion

The aim of the relation conversion is to get the target object geographical information in world coordinate where its center is the base of robot arm. According to 2.2, obtaining the external camera parameters is the key to solve the geographical relation conversion. Review that the conversion from pixel coordinate to world coordinate from Zhang's paper [7]:

$$Z\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a & G & u0 & 0 \\ 0 & b & v0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

After the matrix transformation [8]:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} a & G & u0 & 0 \\ 0 & b & v0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}^{-1} Z\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

Where X, Y and Z are geographical coordinate information, $\begin{bmatrix} a & G & u0 & 0 \\ 0 & b & v0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$ is the camera intrinsic matrix and $\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}$ is the Camera Extrinsic matrix while R is the rotation matrix from camera to the center of world and T is the relative displacement from camera to center of world.

Now, I can get the target object coordinate information in world coordinate. The next step is to let manipulator approach the target object, Actionlib and Moveit should be introduced.

2.4.2 Actionlib and Moveit

This part is to discuss how to control and plan the manipulator operation. In my manipulator control system, Actionlib and Moveit are used to communicate with robot arm and control it. Actionlib is a library which contains various convenient interfaces for user. It is easy to build connection with server to send action request using Actionlib. Moveit is a functional software platform helps users plan manipulator motion and calculate trajectory easily. In my project, three motion planning methods will be used to plan manipulator trajectory path and move manipulator.

Actionlib[26]

Actionlib is an important package in ROS, it can help users control robot to reach the target position easily, in the process, some feedback information can be obtained, interruption can be controlled. The client-server communication mode is used in Actionlib. Actionclient and actionserver communicate through "ROS action Protocol", which is transmitted by ROS message. In addition, actionclient and actionserver provide users with some simple interfaces. Users can use these interfaces to complete the goal request (client side) and goal execution (server side).

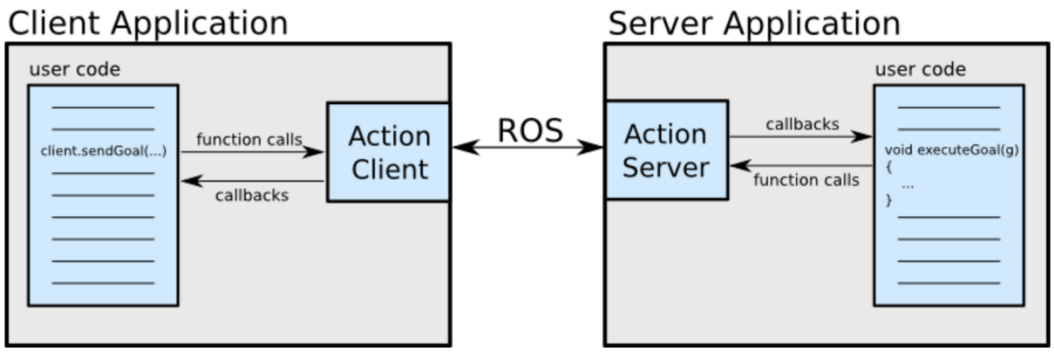


Figure 11. client-server action protocol

Action protocol is used to communicate between actionclient and actionserver. Action protocol is a predefined set of ROS messages. These messages are put on ROS topic to communicate between actionclient and actionserver

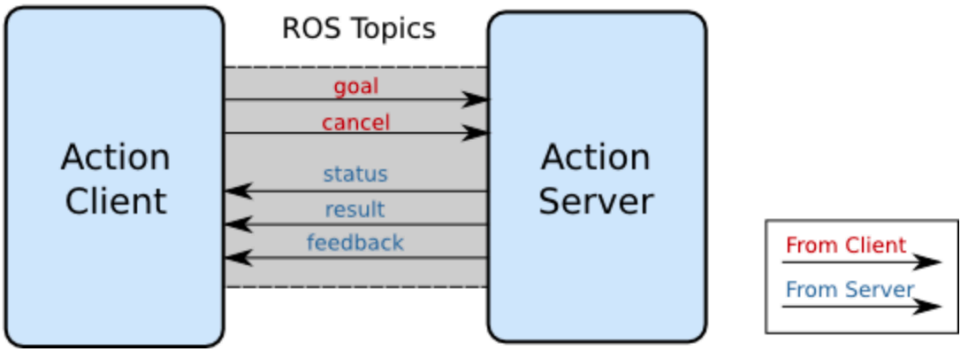


Figure 12. ROS message for client and server

The purpose of goal message is to send the goal information likes position, velocity and rotation information to action server, in my project, the content in goal message is geometry_msgs::Pose. Cancel message is transformed from client to server to send the cancel request to action server. Statues message is aiming to send client goal status information and result message can return the execution results of goal target and feedback message can be auxiliary information.

Moveit[27]

Moveit is the most widely used software platform for manipulation which contains lots of functional library including motion planning, manipulation, kinematics, collision detection and so on. The core node of Moveit is move_group, pulling all the individual components together to provide a set of ROS action and service for user to use. Besides, there are various robot interfaces. Move_groupc can communicate with the robot through ROS topics and actions. It can obtain the current status of the robot (such as joint position), obtain point cloud or other sensing data, and communicate with the robot controller.

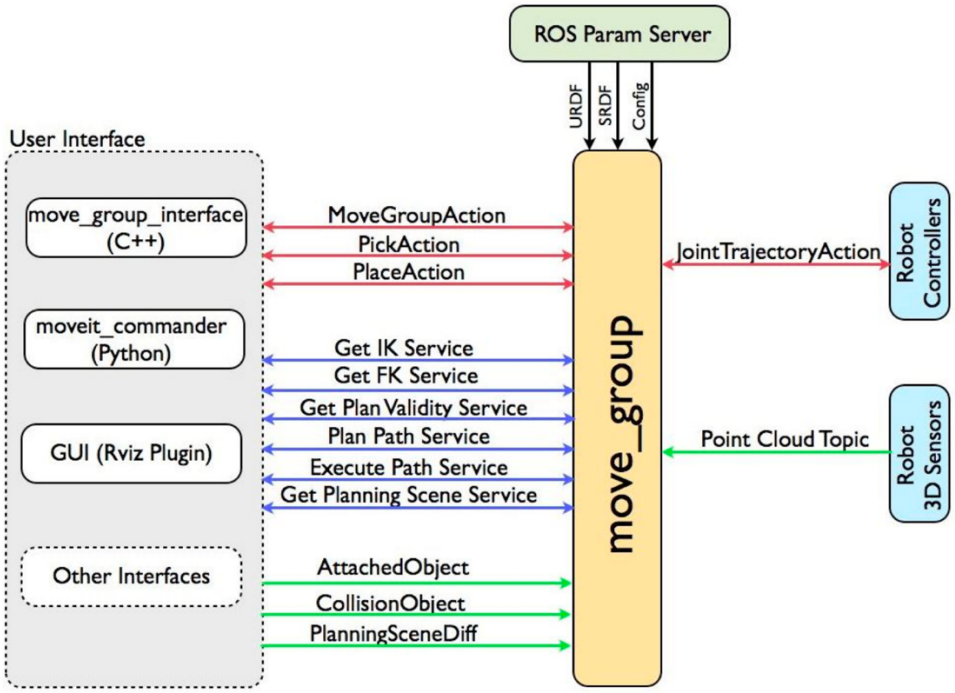


Figure 13. communication through move group node

In my project, moveit_commander interface is used to control robot arm and get access to IK service, Get Plan Validity Service, Plan Path Service, Execute Path Service. Also, move group can get robot and scene config through URDF and SRDF file.

There are three main different methods for controlling manipulator that access to users:

1. Setting Joint Goal

As the name of the method says, it is mainly realized by calling the get_current_joint_value function in robot_commander library. In this mode, the angle of each joint of manipulator can be set and parameters will be sent to serve. Then, manipulator adjusts the angle of each joint in current states. However, when the robot arm is sent an unreasonable angle, the movement of the robot arm is prone to problems. Therefore, it is necessary to accurately calculate the target angle of the manipulator in this way.

2. Setting pose goal

This method is mainly realized by calling the set-pose-target function in robot_commander library. In this mode, firstly, the three-dimensional world coordinate based on base of manipulator should be set by set_pose_reference_frame function. Then, the orientation and position information of end-effector can be set and end-effect can move to target position and orientation. However, the trajectory is the main problem in setting pose goal. The selection of trajectory needs to be accurately calculated to improve feasibility and avoid collision and when the position setting exceeds the arm length radius, the arm will become deadlock status.

3. Setting Cartesian path

It is important that in cartesian path the end-effector can move along string and the arm can avoid collision. This method is mainly realized by calling the computer-cartesian-path function in robot_commander library. Trajectory can be set by specifying a list of waypoints for end-effector to go through. It is very friendly that the moveit-commander provides the function package of calculating Cartesian path and functions to set waypoints conveniently.

In my manipulator control system, I set joint angle to control the opening and closing state of grasper and set the waypoints of Cartesian path to control manipulator to reach specified position.

2.4.3 Inverse Kinematics Solver [28]

The Inverse Kinematics solver is to help the user calculate the angle of the manipulator in joint space corresponding to the Cartesian space position.

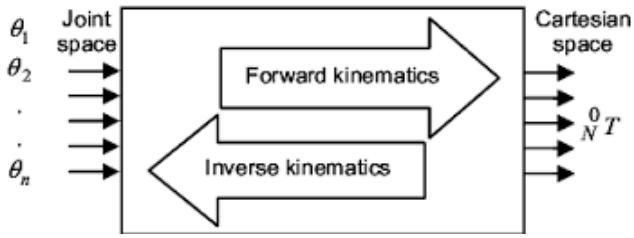


Figure 14. relationship between forward kinematics and inverse kinematics

Trac-IK solver which is supported by ROS platform will be used in my project to calculate the inverse kinematics to get the various angles value of each manipulator joints.

Kinematics and Dynamics library is very popular in the field of industrial robots. KDL provides various class libraries for geometrical objects, kinematic chains of various families and their motion specification and interpolation. Besides, KDL support the transformations of vectors, points, frame and support computation from forward position kinematics, to inverse dynamics.

Like KDL, trac-IK is also a Robot Kinematics solver based on numerical solution, but it has made many improvements on the algorithm level. When there are joint limits which is common for many robotic platform, KDL's convergence algorithms which is based on Newton's method does not work well. Besides there are two IK implementations running in TRAC-IK algorithm. One is a simple extension to KDL's Newton-based convergence algorithm that detects and mitigates local minima due to joint limits by random jumps. Another is a Sequential Quadratic Programming nonlinear optimization method. Therefore, Compared with KDL, the solution efficiency (success rate and calculation time) is much higher.

2.4.4 Motion Planner [28]

After getting the manipulator angle information in joint space, the next step is to plan the manipulator motion. The Open Motion Planning library which is supported by MOVEIT platform is used in project. Open motion planning library is a powerful collection of the latest motion planning algorithms based on sampling and it is the default planner in MOVEIT. Several planners that are part of OPML planning library are capable of optimizing for a specified optimization objective and the geometri::PRTstar is used for the optimal planners.

2.5 Project Equipment

In this section, I will introduce all equipment I need for project design, including Universal robot arm, Kinect RGB-D camera, Robotiq-85 gripper and their device parameters.

2.5.1 Universal Robot Arm

Universal robot manipulator is a 6-DOF manipulator which has good compatibility with ROS and has good extension performance at the end of manipulator.

Here are some important data [29] about Universal robot manipulator:

Degree of freedom	6
Working radius	300mm
Shoulder pan working degree	+- 360 deg
Shoulder lift working degree	+- 360 deg
Wrist 1 working degree	+- 360 deg
Wrist 2 working degree	+- 360 deg
Wrist 3 working degree	+- 360 deg

Table 1. part of UR manipulator data sheet

It is well worth mentioning that the working radius of Universal robot arm is 300mm, therefore, the placed objects should be placed smaller than the working radius.

2.5.2 Kinect RGB-D camera

Kinect is developed by Microsoft and used in peripherals of Xbox 360 and Xbox One consoles. Kinect is widely used in industrial recognition because it has a low price and can guarantee the quality of the depth and RGB images obtained from the camera. In my project, Kinect V2 is used to recognize object and get depth information about object.Here are Kinect V2 useful data table [30] below:

RGB camera resolution	1920x1080
Depth camera resolution	512x424
Depth camera angle (Vertical)	60deg
Depth camera angle (Horizontal)	70deg
Range of Detection	0.5~4.5m

Table 2. Part of Kinect V2 data sheet

It is well worth mentioning that the range of detection of Kinect is from 0.5 to 4.5 meter, therefore, the placed objects should be in the detection range and the resolution in depth space is 512x424 which will be used in identifying object part.

2.5.3 Robotiq-85 Gripper [31]

Robotiq-85 gripper is a plug and play gripper on a cooperative robot. Robotiq-85 gripper is used to grasp target object because the two fingers gripper with a distance 85cm is very suitable for grasping small object in my project, besides, Robotiq-85 gripper has good compatibility on Universal Robot manipulator.

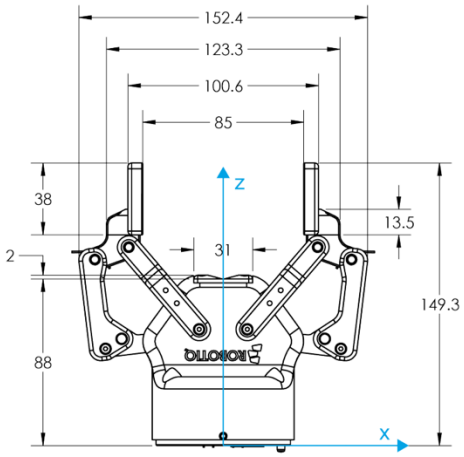


Figure 15. Robotiq-85 gripper

The distance between two fingers is 85cm, therefore, it is very suitable for small object grasping. The depth distance information will be used to solve the placement problem of pre-grasping. As we can see in figure 14, the depth of robotiq-85 gripper is 40 mm and the width of robotiq-85 gripper is 85 mm, therefore, the three dimensions information target object should be set within the range.

3. My work

After the above analysis of all theoretical knowledge. In this part, I will show each designing step and results of project in details including configuring working environment, building manipulator grasping system, target object recognition scheme, manipulator grasping. Some results I get will be analyzed and compared with another result. Here is the whole manipulator grasping system flowchart:

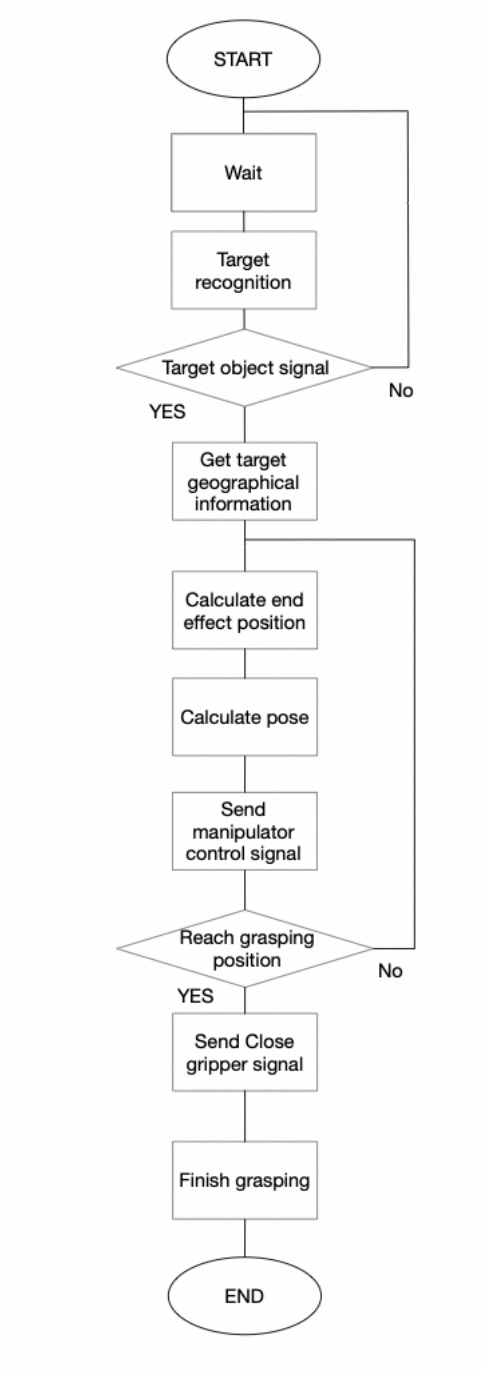


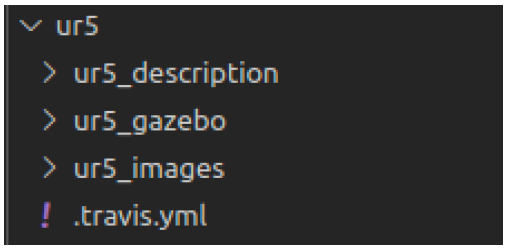
Figure 16. grasping process flowchart

3.1 Working Environment

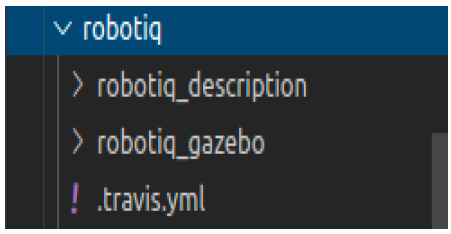
Working environment includes hardware environment and software environment. My work is done in Ubuntu 16.04 built by Parallel virtual machine in MacBook Pro (13-inch, 2019) with i5 CPU and 16 GB 2133Mhz memory, hardware environment. As for software environment, the ROS vision is Kinetic and the whole system is simulated in a simulation modeling software called Gazebo, the manipulator grasping system status can be monitored in real time in Rviz.

3.2 Building Manipulator Grasping System

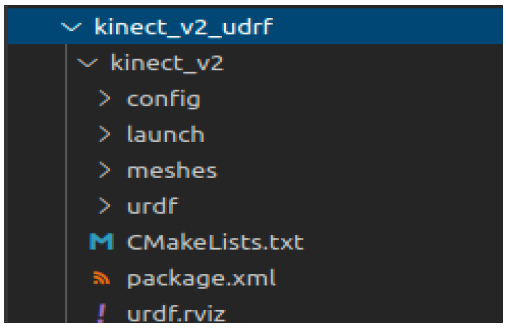
The whole manipulator grasping system is simulated in a software Gazebo and there are three open source code packages from GitHub applied in project.



(a) [36]



(b)[37]



(c)[38]

Figure 17. three open source code packages from GitHub

There are urdf files in Ur5_description in ur5 package that contains some source code to build Ur5 manipulator model without gripper.Robotiq package contains gripper model code information which is described in robotiq_description file.What is more, the Kinect V2 model is described in Kinect-v2-urdf package. Then, these three description packages should be merged into one macro file to launch in Gazebo.

Here is the macro file which can combine there three description model packages together, named ur5-robotiq85-gripper.urdf.xacro file .

```
<!--add ur5 model/-->
<xacro:include filename="$(find ur5_description)/urdf/ur5_joint_limited_robot.urdf.xacro" />
<!--add robotiq_gripper/-->      You, a few seconds ago • Uncommitted changes
<xacro:include filename="$(find robotiq_description)/urdf/robotiq_85_gripper.urdf.xacro" />
<!--add a kinect depth camera/-->
<xacro:include filename="$(find franka_gazebo)/robots/sensor/kinect_v2.urdf.xacro" />
  <!--link name="world"/-->
  <xacro:kinect_v2  parent="world">
    <origin xyz="0.4 0 1"
            rpy="0 1.57 0" />
  </xacro:kinect_v2>
```

Figure 18. Part of ur5-robotiq85-gripper.urdf.xacro file

In this part of code, the Kinect camera is placed in the position of (0.4, 0, 1) from the world base center point and Euler angle turns 180 degrees anticlockwise with respect to the P direction of origin. Here is the manipulator-camera system without target object:

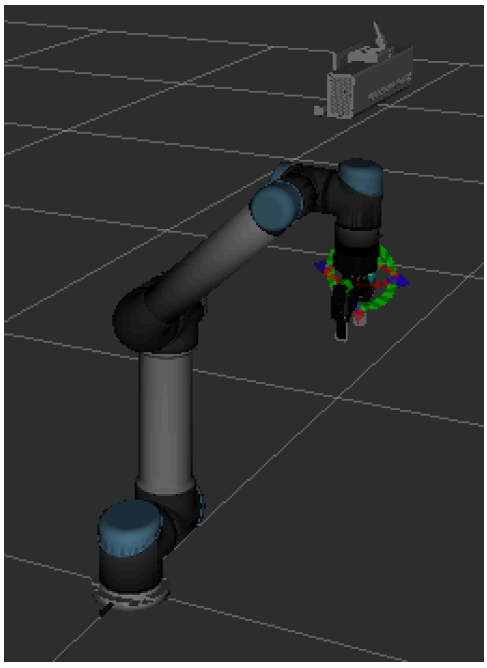


Figure 19. Manipulator-camera system

The next step is to add the target object which will be recognized by camera and grabbed by manipulator. After analyzing the parameter relationship between manipulator, gripper. A table, a bench and three colored squares will be added into world.

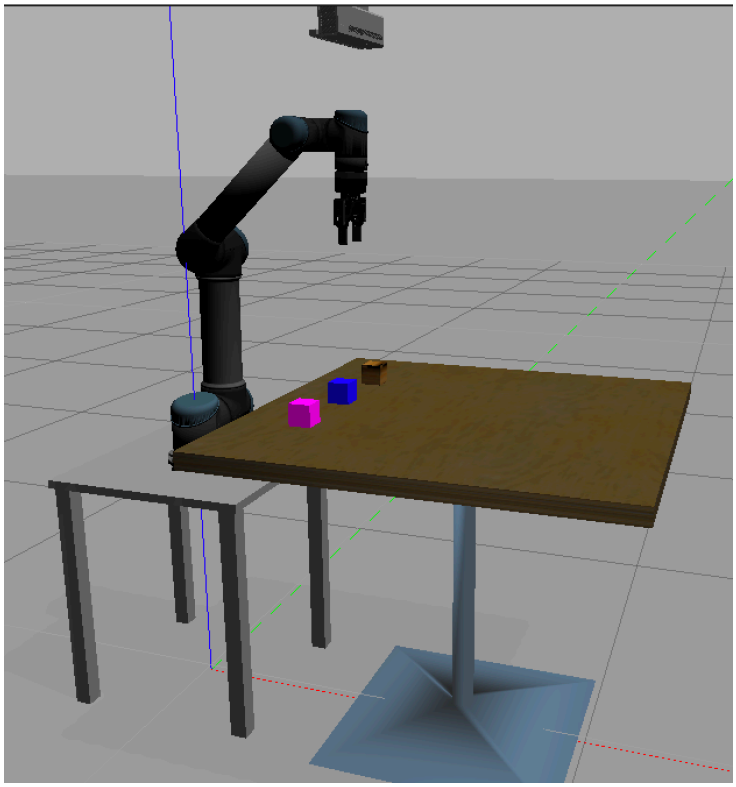


Figure 20. the whole manipulator grasping system

the café table is placed to the position (0.7, 0, 0) and the height of café table is 0.775m.

the bench is place to the position (0, 0, 0) and the height of bench is 0.58m.

three cubes are place to the café table (0.4, 0.2), (0.4, -0.2), (0.4, 0) respectively.

Now, it is the time to recognize target color cube by Kinect camera and make manipulator grasp it. The rgb and depth pictures gotten from Kinect camera are as follows:

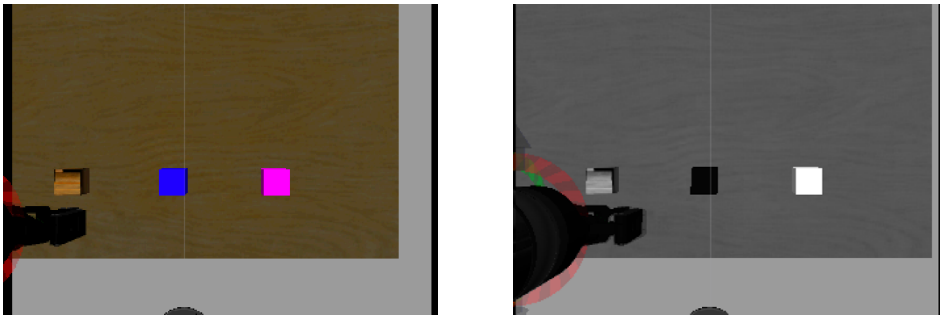


Figure 21. rgb and depth image from Kinect v2 camera

3.3 Target Object Recognition Scheme

In my project, I decide to use two different target object recognition methods: SIFT recognition and OpenCV method to get the geographical position information.

Find-object

As for SIFT recognition, there is a package called find-object-2d which is a powerful function package to recognize object in ROS platform. Due to the relatively few feature points of pure color object, therefore, I decide to choose wood as the surface material which can provide target object with more feature points.

Firstly, some find-object parameters need to be configured. Fortunately, find-object-2d is very friendly to Kinect camera. There is a version called find-object-3d-kinect specially released for Kinect camera.

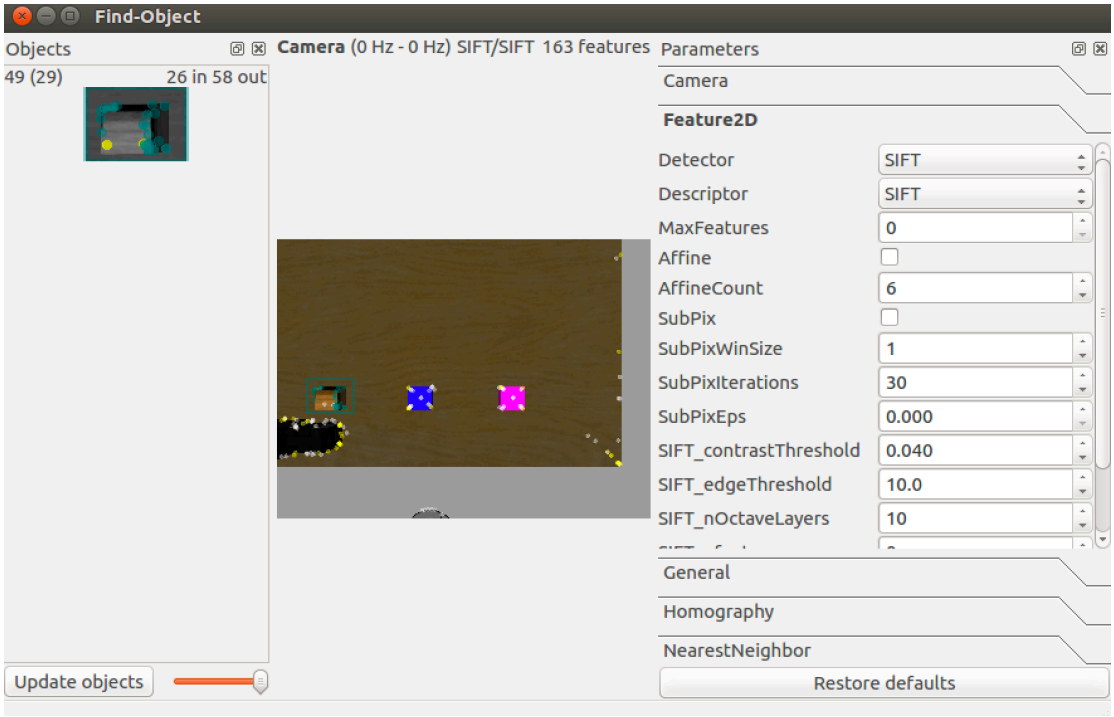


Figure 22. the find-object-3d-kinect window

According to SIFT principle, there should be a target image as the initial image to calculate the keypoints by derivative of Gaussian.

On the left side of window shown above, the initial image should be added and the target will be labeled when there is enough keypoints of target.

As we can see on the right side of window. the SIFT_nOctavelLayers is set as 10, which means the number of layers in octave is 10 (3 is the value used in D.Lowe's

paper). The number of feature points is larger than default config when the layers of each octave is 10. Besides, the Gaussian smooth coefficient is set as 1.6 which is the sigma of Gaussian applied to the input image at octave (Hint: the larger the Gaussian smooth coefficient is, the more blurred the image is).

After detecting the target object by camera, the target object geographical topic information should be published. By reading the find-object source code, I can obtain that there is a new target frame in the Kinect v2 coordinate:

Figure 23. the TF free when target is detected

Then, it is easy to get the geographical information of target in the camera coordinate, the most convenient method is the tf_echo method.

```
At time 338.001
- Translation: [-0.304, 0.040, 0.787]
- Rotation: in Quaternion [0.499, 0.502, -0.501, 0.498]
            in RPY (radian) [-1.701, 1.565, 3.011]
            in RPY (degree) [-97.464, 89.666, 172.533]
```

Figure 24. the position and rotation information of target in camera coordinate

Finally, in order to grasp target conveniently, it is necessary to transform the geographical target information from camera coordinate into manipulator coordinate centered on the manipulator base.

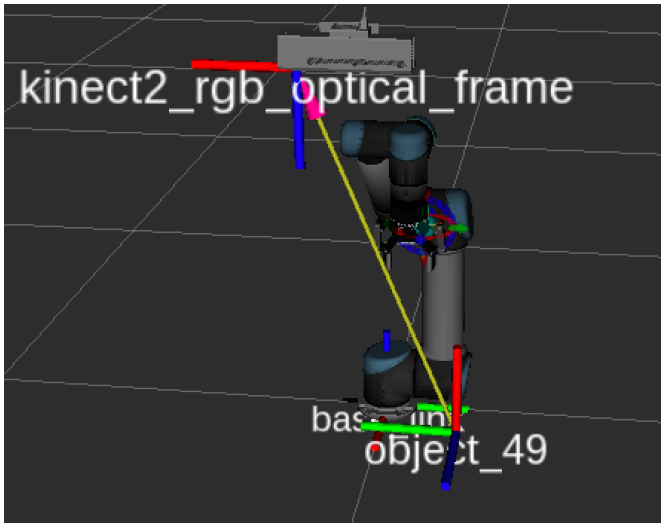


Figure 25. relationship among object, base link and optical frame

As we can see above, the object was detected in optical frame coordinate by Kinect camera. Therefore, in this step, a tf_listener which can listen the target coordinate transformation process and publish a new topic to ROS topic manager should be built by python coding.

```python
#!/usr/bin/env python

import rospy
import tf
from geometry_msgs.msg import Pose
def object_position_pose(t,o):
    pub = rospy.Publisher('/objection_position_pose',Pose,queue_size=10)
    p = Pose()
    rate = rospy.Rate(5)
    p.position.x = t[0]
    p.position.y = t[1]
    p.position.z = t[2]

    p.orientation.x = o[0]
    p.orientation.y = o[1]
    p.orientation.z = o[2]
    p.orientation.w = o[3]
    pub.publish(p)
    rate.sleep()

if __name__ == '__main__':
   rospy.init_node('tf_listener',anonymous=True)
   listener = tf.TransformListener()
   rate = rospy.Rate(10.0)
   while not rospy.is_shutdown():
       try:
           (trans,rot) = listener.lookupTransform('/world', '/object_34', rospy.Time(0))
           print("trans:")
           print(trans)
           print("rot:")
           print(rot)
           object_position_pose(trans,rot)
       except (tf.LookupException, tf.ConnectivityException, tf.ExtrapolationException):
           continue
```

Figure 26. tf_listener.py file

As we can see, the aim of tf-listener.py file is to get the position and orientation information in world coordinate. In main function, listener.lookupTransform will

complete the calculation. It is well worth to mention that there is a topic called objection_position_pose published in function object_position_pose, the message format of topic objection_position_pose is Pose message.

```
hongshaorou@ubuntu:~/ros_ws$ rosmsg show geometry_msgs/Pose
geometry_msgs/Point position
  float64 x
  float64 y
  float64 z
geometry_msgs/Quaternion orientation
  float64 x
  float64 y
  float64 z
  float64 w
```

Figure 27. contents of Pose message

Pose message contains two different kinds of information, position information and orientation or Quaternion information. In tf_listener.py file, the position information is the relative position of target in world frame and orientation is the relative quaternion information in world frame. Finally, here is the results.

```
hongshaorou@ubuntu:~/ros_ws$ rosrun opencv tf_listener.py
trans:
[0.40262515932595755, 0.20887835802307397, 0.1807008788096327]
rot:
[-0.7065538628825438, -0.002062721022281591, -0.7076533700631031, -0.00202283614
57967758]
```

Figure 28. results gotten from find-object package

OpenCV

The second method is OpenCV, the target object will be recognized by processing the image with OpenCV and the particle of target can be calculated. The flow chart of OpenCV method is as followed:

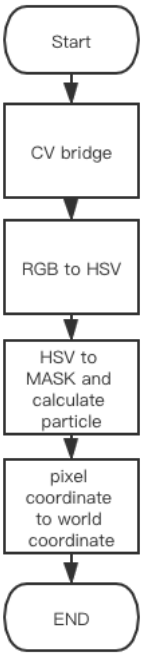


Figure 29. OpenCV method flow chart

The main idea of OpenCV method is to transform RGB color model which is widely applicable to color monitors and color video cameras to HSV color model which are more in line with the way people describe and explain colors. HSV color descriptions are more intuitive for people.

RGB [32] means red, green, blue, these three parameters correspond to brightness. RGB model is a three-dimensional coordinate system model in Cartesian space and it is additive. HSV [33] means Hue, Saturation and Value. HSV color model is a Hexcone model [34] created by A. R. Smith in 1978.

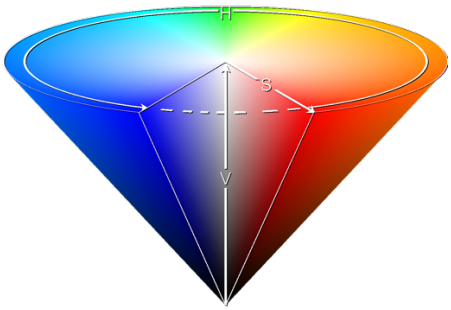


Figure 30. HSV color Hexcone model

Hue is described by degree and ranging from 0 to 360 degree. HSV model counters clockwise from red. 0 degree represents red color, 120 degree represents green color and 240 degree represents blue color.

Saturation represents how close the color is to the spectral color and the range of Saturation is from 0% to 100%. The higher saturation is, the deeper and more vivid the color is. Value parameter is the brightness of the color and usually the Value range is 0% (black) to 100% (white).

Firstly, in ROS platform, it is convenient to get the image format that OPENCV can understand by using cv-bridge which is a useful interface to transform the ROS image message format to OpenCV format. In my project, the picture gotten from Kinect camera is sensor_msgs/image format should be transformed to cvimage.

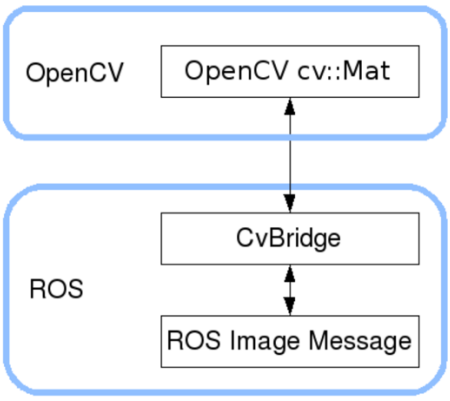


Figure 31. relationship between OpenCV, ROS and CvBridge.

The picture above is the relationship between OpenCV, ROS and CvBridge. In a and b, c acts as a bridge to connect communication between ROS and OpenCV.

```
18  class image_converter:
19    def __init__(self):
20      self.image_pub = rospy.Publisher("/Kinect_V2/ir/image_raw_converter",Image,queue_size=10)
21      self.bridge = CvBridge()
22      self.image_sub = rospy.Subscriber("/kinect_V2/rgb/image_raw",Image,self.callback)
23    def callback(self,data):
24      try:
25        cv_image = self.bridge.imgmsg_to_cv2(data, "bgr8")
26      except CvBridgeError as e:
27        print(e)
28
29      (rows,cols,channels) = cv_image.shape
30      cv2.imshow("Image window", cv_image)
31      cv2.waitKey(3)
32
33      try:
34        self.image_pub.publish(self.bridge.cv2_to_imgmsg(cv_image, "bgr8"))
35      except CvBridgeError as e:
36        print(e)
```

Figure 32. Image-converter from ROS image to OpenCV image

After getting the CVimage, the image can be processed by OpenCV. The next step is to transform RGB color model to HSV color model. It can be conveniently achieved by:

```
43      try:
44          cv_image = self.bridge.imgmsg_to_cv2(data, "bgr8")
45      except CvBridgeError as e:
46          print(e)
```

Figure 33. CVbridge function

Then, the target color should be unchanged and others should be filtered out which mean the image I got after filtering is white and black. Besides, a certain particular weighted average intensity of image pixel should be calculated and the corresponding points should be marked in the image.

Here are related codes:

```
51      lower_wood = np.array([10,43,46])
52      upper_wood = np.array([18,255,255])
53      # Threshold the HSV image to get only white colors
54      mask = cv2.inRange(hsv, lower_wood, upper_wood)
55
56      (rows,cols,channels) = cv_image.shape
57      # Calculate centroid of the blob of binary image using ImageMoments
58      m = cv2.moments(mask, False)
59      try:
60          cx, cy = m['m10']/m['m00'], m['m01']/m['m00']
61      except ZeroDivisionError:
62          cy, cx = rows/2, cols/2
63      # Draw the centroid in the resultut image
64      # cv2.circle(img, center, radius, color[, thickness[, lineType[, shift]]])
65      cv2.circle(mask,(int(cx), int(cy)), 10,(0,0,255),-1)
66      print(cx,cy)
67      cv2.imshow("HSV", hsv)
68      cv2.imshow("MASK", mask)
69
70      cv2.waitKey(1)
```

Figure 34. RGB to HSV and image moments calculation

In code, lower_wood parameter and upper_wood parameter are the HSV filtering range of wood color. The filter function can be easily achieved by using inRange of CV2 function with three inputs. What is more, image information about rows, cols and channels can be gotten from cv_image.shape function. It is well worth mention that the center of image moment [35], the particle of target object is equal to:

$$x_0 = \frac{m_{10}}{m_{00}}; y_0 = \frac{m_{01}}{m_{00}}$$

Finally, the calculated results x0 and y0 should be marked in the image:

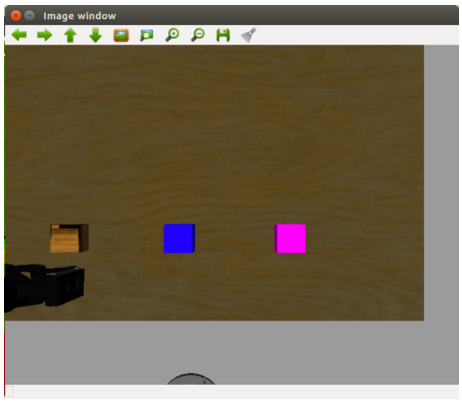


Figure 35. RGB image from camera

Figure 36. HSV image

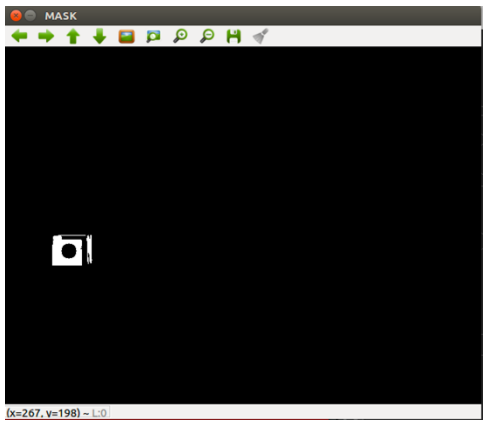


Figure 37. Mask image and center of target object

Now, I can get the center of target object. A topic needs to be created, x0, y0 need to be published to ROS topic manager and the calculation about conversion from pixel coordination to world coordination need to be done.

```
^Chongshaorou@ubuntu:~/ros_ws$ rosrun opencv image_converter.py
86.8647686833 274.264531435
86.8647686833 274.264531435
86.8647686833 274.264531435
86.8647686833 274.264531435
86.8647686833 274.264531435
```

Figure 38. center information in pixel coordinate

Therefore, the pixel coordinate information x0 and y0 are (86.86476,274.2645) which needs to be transformed to world coordinate information.

I can easily get the geographical relationship between camera RGB optical frame and world base frame by using tf_echo function which I have already introduced in find-object part.

According to matrix conversion in background part, pixel coordinate is equal to the dot product of camera intrinsic matrix parameters, camera external matrix parameters and world coordinate.

Review the matrix conversion equation:

$$Z\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a & G & u0 & 0 \\ 0 & b & v0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

The camera intrinsic matrix parameters can be gotten from;

```
hongshaorou@ubuntu:~/ros_ws$ rostopic echo /kinect_V2/rgb/camera_info
header:
  seq: 0
  stamp:
    secs: 15
    nsecs:   1000000
  frame_id: "kinect2_rgb_optical_frame"
height: 480
width: 640
distortion_model: "plumb_bob"
D: [1e-08, 1e-08, 1e-08, 1e-08, 1e-08]
K: [589.3671569219632, 0.0, 320.5, 0.0, 589.3671569219632, 240.5, 0.0, 0.0, 1.0
]
R: [1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0]
P: [589.3671569219632, 0.0, 320.5, -0.0, 0.0, 589.3671569219632, 240.5, 0.0, 0.
0, 0.0, 1.0, 0.0]
binning_x: 0
binning_y: 0
roi:
  x_offset: 0
  y_offset: 0
  height: 0
  width: 0
  do_rectify: False
```

Figure 39. Kinect RGB camera information

Now, the Kinect transform matrix, Kinect rotation matrix and Kinect camera intrinsic matrix are obtained. The calculation process is showing below.

```
5  def pixelcoord2worldcoord(K,matrix_ex,pixel):
6      slice=np.mat([0,0,0])
7      splice = np.zeros((3,1))#create 3x1 zero matrix
8      pixel_T = pixel.transpose()
9      print("pixel_T:")
10     print(pixel_T)
11     K = np.hstack((K,splice))
12     print("the K value after splicing")
13     print(K)
14     # matrix is dot of intrinstic and external parameter
15     matrix = np.dot(K,matrix_ex)
16     print("matrix:")
17     print(matrix)
18     matrix_I = matrix.I
19     print("inverse matrix:")
20     print(matrix_I)
21     # worldcoord = k_I*pixel*R_I -t*R_I
22     worldcoord = matrix_I*pixel_T
23     worldcoord = worldcoord/worldcoord[3]
24     return worldcoord
```

Figure 40. pixel to world coordinate function

Then, the result of matrix conversion is showing below:

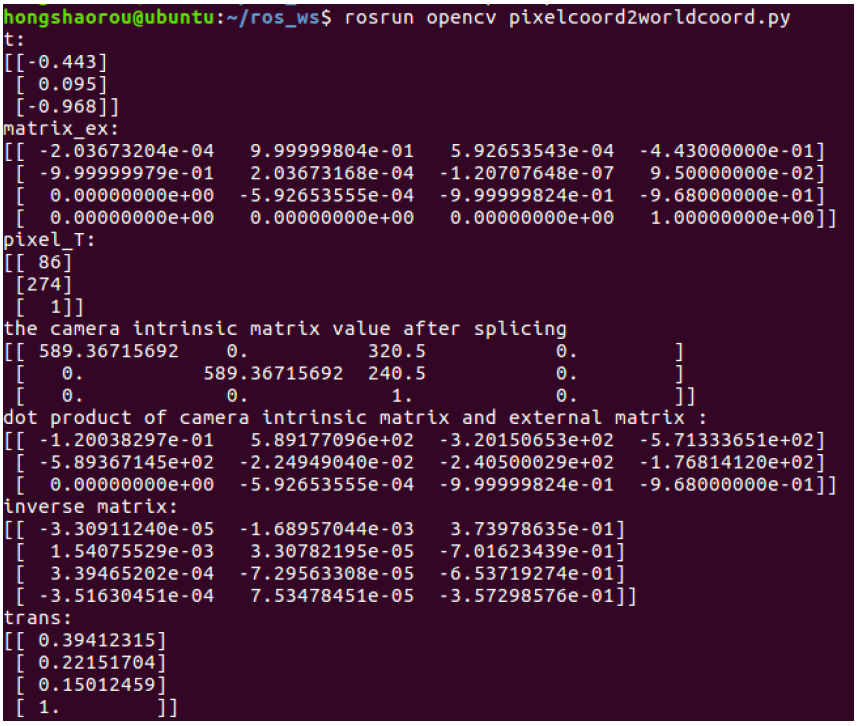


Figure 41. the results matrix gotten from matrix conversion process

As we can see above, I can get the transform matrix (the relative coordinates of objects in Cartesian space system centered on base link) from Kinect camera intrinsic matrix, Kinect rotation matrix and the Kinect transform matrix. The results gotten from OpenCV method is:

[0.39412315, 0.22151704, 0.15012459]

So far, I have used two different methods, SIFT algorithm in find-object package and OpenCV method (identifying object in HSV color model and transform coordinate matrix) to identify objects and gotten two results. In the next section, I will compare the two methods and choose the best result to assist manipulator grasping.

3.4 Comparison of Find-object and OpenCV

So far, I have come to two results in two different methods. In this section, I will compare two results and choose the optimal one.

The error comparison data table is shown below:

Method	Coordinate value	Error in X Y Z			Average Error
Real coordinate in Gazebo	(0.4, 0.2, 0.18)	0	0	0	0
SIFT method	(0.40202516, 0.208878358, 0.180700878)	0.0050629	0.00887835	0.00389372	0.00594
OpenCV method	(0.39412315, 0.22151704, 0.15012459)	0.00587685	0.02151704	0.02987541	0.01909

Table 3. comparison between find-object and OpenCV

As we can see in the table above, the error caused by OpenCV method is obviously greatly than the error caused by Sift method. The errors of two methods in X axis are almost same, 0.5% different from the real one. However, the SIFT method is much better than the OpenCV method in Y and Z axis. There is only 0.00888 error in Y axis and 0.00389 error in Z axis for Sift, while there is 0.0215 error and 0.03 error in Y and Z axis respectively for OpenCV method. In addition, the average error of second method is four times larger than the Sift method' s.

Therefore, after comparing two methods error, the three-dimensional coordinate information generated by Sift method should be chosen according to its lower error and more accurate data.

3.5 Manipulator Grasping

In this section, I will integrate the whole previous knowledge to let the manipulator perform the grasping work. The main steps are shown in the following flowchart:

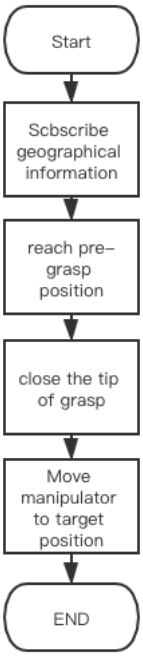


Figure 42. flowchart about grasping

First of all, the choice of controlling mechanism for manipulator should be discussed. In motion planning section, I have listed three types of ROS controlled manipulator interface and they are setting joint goal, setting pose goal and setting Cartesian waypoints respectively.

In my project, the setting Cartesian waypoints will be selected to control the manipulator movement because collisions can be avoided by making the end effect of manipulator move straight by waypoints algorithm, while the movement path cannot be controlled by setting pose goal and setting the angle of each joint in not intuitive for user.

In target object recognition scheme section, a tf-listener was created to listen target object geographical information and publish a new topic to ROS manager. Therefore, it is important to subscribe the new topic from ROS manager to get the target geographical information after coordinates transform.

```
/objection_position_pose
/objects
/objectsStamped
```

Figure 43. Three important ROS topics

These three topics are important, /objects and /objectsStamped contains the geographical information before matrix conversion. Topic /objection_position_pose contains three-dimensional geographical information about target object after coordinate conversion. Therefore, /objection_position_pose should be subscribed if I want to know target information. After subscribing /objection_position_pose topic, Callback function should be modified:

```
18  def callback(pose):
19      object_position_info = pose.position
20      object_orientation_info = pose.orientation
21      print object_position_info
22      moveit_commander.roscpp_initialize(sys.argv)
23      #rospy.init_node('moveit_cartesian', anonymous=True)
24      cartesian = rospy.get_param('~cartesian', True)
25
26      #set cartesian parameters
27      ur5_manipulator = MoveGroupCommander('ur5_manipulator')
28      ur5_gripper = MoveGroupCommander('ur5_gripper')
29      ur5_manipulator.allow_replanning(True)
30      ur5_manipulator.set_pose_reference_frame('base_link')
31      ur5_manipulator.set_goal_position_tolerance(0.01)
32      ur5_manipulator.set_goal_orientation_tolerance(0.1)
33      end_effector_link = ur5_manipulator.get_end_effector_link()
34      ur5_manipulator.set_named_target('ready')
35      ur5_manipulator.go()
36      ur5_gripper.set_named_target('open')
37      ur5_gripper.go()
```

Figure 44. Callback function in grasp.py

The message in /objection_position_pose topic is Pose format, therefore, I can get the position and orientation information of object in /objection_position_pose. Besides, it is important to set MoveGroupCommander which is a interface of MoveGroup. As we can see above, I set two MoveGroupCommander, manipulator and gripper. Some cartesian path parameters should be set, such as reference frame, position tolerance and orientation tolerance. Except for these parameters, some joint states can be easily set by Moveit setup assistant, a convenient tool for setting manipulator configuration and SRDF file.

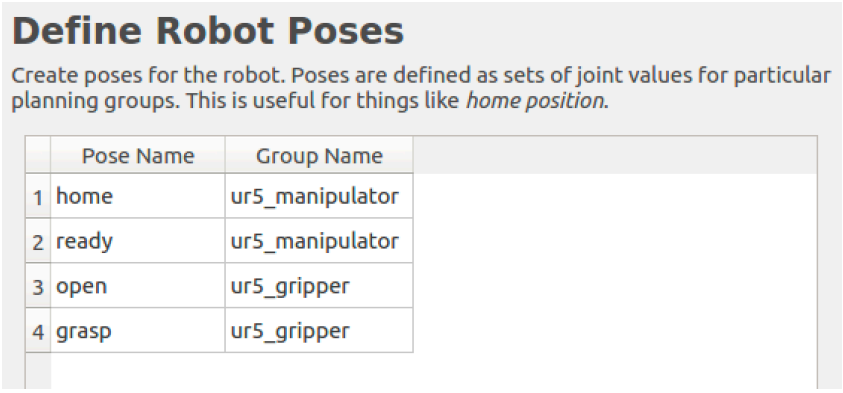


Figure 45. Preset Robot Poses

As we can see, there are four robot poses preset by moveit setup assistant, home and ready states for manipulator, open and grasp states for gripper. The angle values for each joint:

```
24      <group_state name="home" group="ur5_manipulator">
25          <joint name="elbow_joint" value="-0.0345" />
26          <joint name="shoulder_lift_joint" value="-1.588" />
27          <joint name="shoulder_pan_joint" value="0" />
28          <joint name="wrist_1_joint" value="0" />
29          <joint name="wrist_2_joint" value="0" />
30          <joint name="wrist_3_joint" value="0" />
31      </group_state>
32          <group_state name="ready" group="ur5_manipulator">
33          <joint name="elbow_joint" value="0.785" />
34          <joint name="shoulder_lift_joint" value="-1.571" />
35          <joint name="shoulder_pan_joint" value="0" />
36          <joint name="wrist_1_joint" value="-0.785" />
37          <joint name="wrist_2_joint" value="-1.571" />
38          <joint name="wrist_3_joint" value="0" />
39      </group_state>
40      <group_state name="open" group="ur5_gripper">
41          <joint name="robotiq_85_left_knuckle_joint" value="0.1" />
42      </group_state>
43      <group_state name="grasp" group="ur5_gripper">
44          <joint name="robotiq_85_left_knuckle_joint" value="0.4" />
45      </group_state>
```

Figure 46. Angle Values for Each Joint

It is well worth mentioning that in order to simplify grasping operation, the end effect of manipulator should be perpendicular to the desktop of target object so that we can only care about the position of object not the rotation of gripper. According to the figure shown above, in the states of ready, the end effect of manipulator is perpendicular to the desktop. Therefore, the initial position of manipulator should be set as ready states.

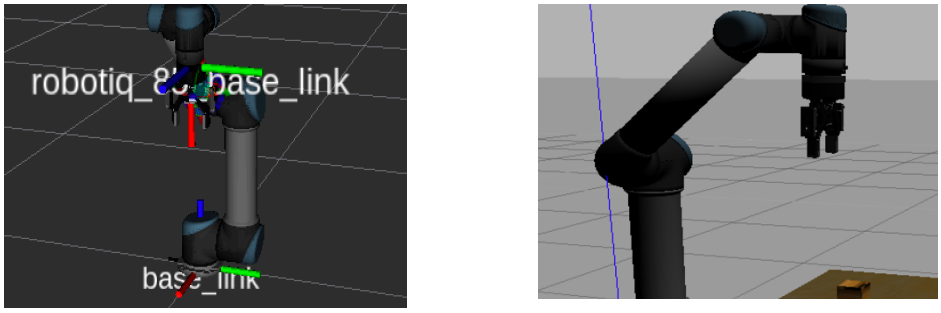


Figure 47. Relationship Between End Effect and Base Link, Ready States for Manipulator

After initializing poses of manipulator, next will be the most critical step. The end effect of manipulator will be moved to the pre-grasp position by calculating Cartesian waypoints.

```
58      if cartesian:
59          fraction = 0.0
60          maxtries = 100
61          attempts = 0
62          while fraction < 1.0 and attempts < maxtries:
63              (plan, fraction) =  ur5_manipulator.compute_cartesian_path (
64                                      waypoints,
65                                      0.01,
66                                      0.0,
67                                      True)
68              attempts += 1
69
70
71              if attempts % 10 == 0:
72                  rospy.loginfo("Still trying after " + str(attempts) + " attempts...")
```

Figure 48. Cartesian Waypoints

Here is the Cartesian Waypoint function, Compute_cartesian_path function, in this function, there are four inputs. Waypoints input is a waypoints array which contains the geographical waypoints information. 0.01 means that the end effect step is 0.01m, a resolution of 1cm will be interpolated to the Cartesian path. And jump threshold will be banned, therefore the third input is 0.0. Besides, True means that collision avoiding is allowed.

It should be noticed that the default inverse kinematic solver is KDL kinematics solver and the success rate of inverse kinematics calculation is very low. Therefore, in my project, the KDL kinematics solver will be replaced by Trac-ik kinematic solver which I have already discussed in inverse kinematic solver part.

It is convenient to replace KDL inverse kinematics solver with trac-ik solver in Kinematic.yaml configuration file.

```yaml
ur5_manipulator:
  kinematics_solver: trac_ik_kinematics_plugin/TRAC_IKKinematicsPlugin
  kinematics_solver_search_resolution: 0.005
  kinematics_solver_timeout: 0.005
  kinematics_solver_attempts: 3
ur5_gripper:
  kinematics_solver: kdl_kinematics_plugin/KDLKinematicsPlugin
  kinematics_solver_search_resolution: 0.005
  kinematics_solver_timeout: 0.005
  kinematics_solver_attempts: 3
```

Figure 49. trac-ik kinematics solver and kdl kinematics solver

Here is the waypoints array:

```python
  #get the end effort information
  start_pose = ur5_manipulator.get_current_pose(end_effector_link).pose
  print("The first waypoint:")
  print(start_pose)
  #define waypoints
  waypoints = []
  waypoints.append(start_pose)

  wpose = deepcopy(start_pose)
  wpose.position.z = object_position_info.z+0.27
  wpose.position.x = object_position_info.x
  wpose.position.y = object_position_info.y
  print("The second waypoint:")
  print(wpose)
  waypoints.append(deepcopy(wpose))
  print(" ")
  print(waypoints)
```

Figure 50. waypoints array

As we can see, waypoints array contains three-dimensional geographical information which represents a point in Cartesian space centered on base of manipulator. Then, the gripper state should be set to grasp states and manipulator will move to the target position when it is stable.

```python
    closeGripper()
    rospy.sleep(2)
    ur5_manipulator.set_named_target('ready')
    ur5_manipulator.go()
    rospy.sleep(3)
```

Figure 51. Closed Gripper and Move Manipulator

The target position of manipulator is set to ready states.

Finally, let's run the grasp.py file and final results will be shown below:

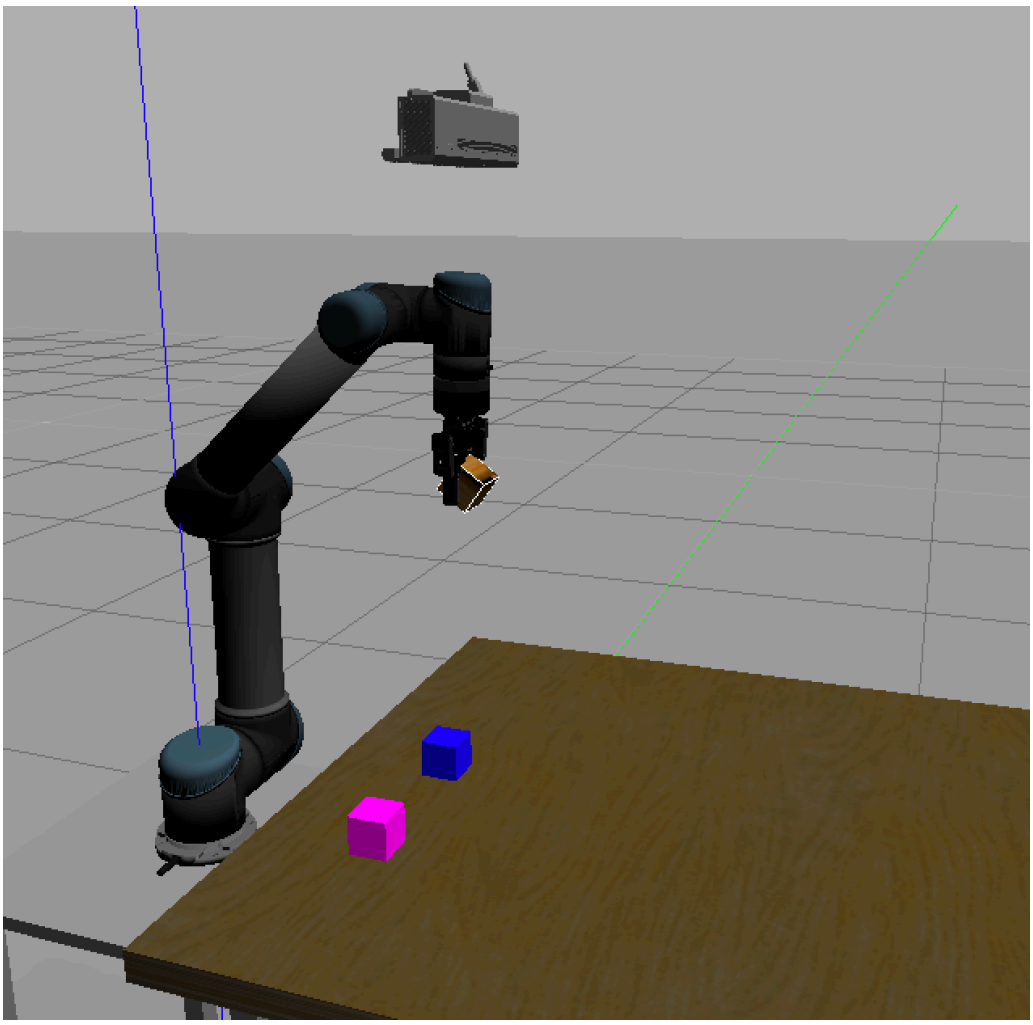


Figure 52. Manipulator grasping results

There are three different colors cubes, wood color, blue and pink colors respectively, the gripper attached to the end effect of manipulator grasp the wood color cube and place it to a target position.

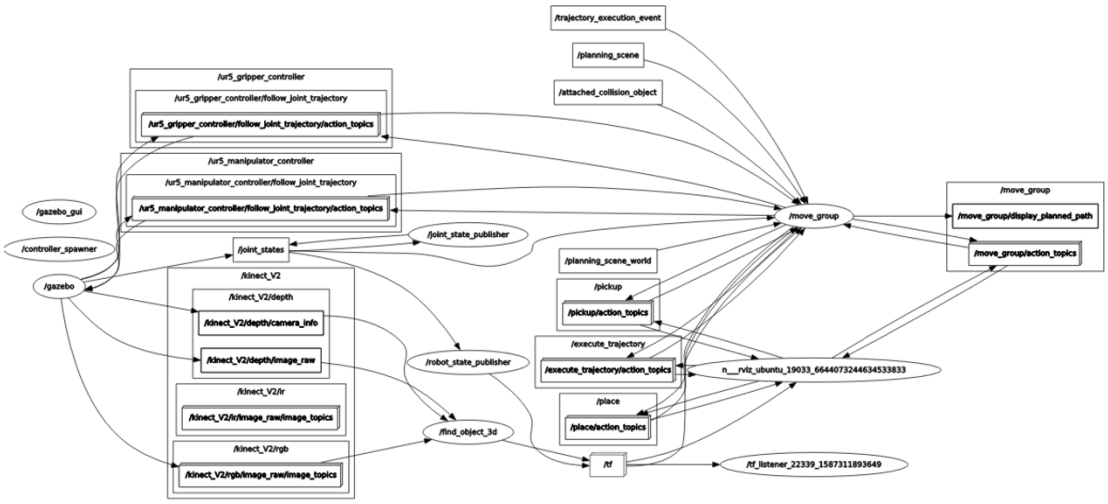


Figure 53. project nodes and topics structure

The oval represents node and rectangle represents topics. The four most important nodes are /gazebo, /tf_listener, /find_object_3d and /move_group. Node gazebo can publish model information and group controllers information. Node tf_listener can subscribe topic tf, monitor each joint state and calculate the coordinate conversion. Node find_object_3d can recognize target by subscribing Kinect V2 RGB and depth topic and publish tf information. Node move_group is the core of controlling manipulator and grasping. All the movement of manipulator and gripper is completed through move_group node.

3.6 problem discussion

So far, I have already achieved the project objective, identifying different color cube and grasping the target color cube to a target place. However, there are some problems or unsatisfactory that should be discussed.

3.6.1 Physical Engines

The physical engines will restrict the performance of manipulator in grasping because the whole project is implemented in the simulation environment. There is difference between the results of simulation in simulator and in real world.

During project completion, I found that when the manipulator gripper grasps the target object, the target can easily slip out of the gripper or fall from the gripper. This is due to a problem with the friction coefficient setting in Gazebo simulator. This problem is improved by adjusting friction coefficient of gripper and cube in .world file generated by Gazebo simulator. However, even if I adjusted the coefficient of friction, there is still a chance that the wood cube will slip or fall from gripper. So, I blamed this problem on the cause of emulator physical engines.

3.6.2 Constraints

In the part of comparison between sift method and OpenCV method, the color of cube is close to the color of table, which affects the accuracy of the results obtained by OpenCV method. Besides, when comparing with two different method by using control variables method, the variables are strict controlled, but there are no enough

experiments. Therefore, when the number of trials increasing, there is a chance of different results.

In addition, the grasping decision of manipulator is controlled by Kinect vision. In the find-object-Kinect platform, it is necessary to match the features of object screenshot with the visual real-time image, so when the object position changes, it is necessary to re-screenshot for matching. Real time grasping needs to be improved.

4. conclusion

This thesis introduces the establishment process of vision-based manipulator grasping system, analyzes sift visual recognition algorithm and compares the applicability of sift algorithm and OpenCV method to the project. Sift recognizes object to calculate coordinates with higher accuracy than OpenCV method.

The physical engines of emulator like frication, have a certain impact on the project results but the results obtained are acceptable. And the comparison between sift method and OpenCV method needs to increase the comparison times to improve accuracy. The real-time grasping for vision-based manipulator grasping system is not supported.

5. reference

[1] W. S. Newsman (2018), “A Systematic Approach to Learning Robot Programming With ROS”, pp. 5.

[2] J. Minichino (2015), “Learning OpenCV Computer Vision with Python” pp 140, [Online] and W. S. Newsman (2018), “A Systematic Approach to Learning Robot Programming With ROS”, pp. 237-245.

[3] W. S. Newsman (2018), “A Systematic Approach to Learning Robot Programming With ROS”, pp. 38-42.

[4] W. S. Newsman (2018), “A Systematic Approach to Learning Robot Programming With ROS”, pp. 9-23.

[5] L. Joseph (2015), “Learning Robotics with Python”, pp 30-35.

[6] Z. Zhang, A Flexible New Technique for Camera Calibration (1998). 1-2

[7] Z. Zhang, A Flexible New Technique for Camera Calibration (1998). Chapter 2.1 Notation, pp 3

[8] Weisstein, Eric W. "Rotation Matrix." From *MathWorld*--A Wolfram Web Resource. [Online] Available: https://mathworld.wolfram.com/RotationMatrix.html

[9] Z. Zhang, A Flexible New Technique for Camera Calibration (1998). Chapter 2.2 Homography between the Model Plane and Its Image, pp 4

[10] Z. Zhang, A Flexible New Technique for Camera Calibration (1998). Chapter 2.3 Constraints on the Intrinsic Parameters, pp 4

[11] Z. Zhang, A Flexible New Technique for Camera Calibration (1998). Chapter 3 Solving Camera Calibration, pp 4-6

[12] Z. Zhang, A Flexible New Technique for Camera Calibration (1998). Chapter B Extraction of the Intrinsic Parameters from Matrix B, pp 18

[13] David G. Lowe Object Recognition from Local Scale-Invariant Features. 1999.

[14] R. Gonzalez and R. Woods, “Digital Image Processing”, Addison-Wesley Publishing Company, 1992, p 173-p179

[15] Shapiro, L. G. & Stockman, G. C: "Computer Vision", page 137, 150. Prentice Hall, 2001

[16] R.A. Haddad and A.N. Akansu, "A Class of Fast Gaussian Binomial Filters for Speech and Image Processing," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 39, pp 723-727, March 1991 and Lowe, David G. (1999). "Object recognition from local scale-invariant features" (PDF). “Proceedings of the International Conference on Computer vision” *2.* Pp 1150-1157

[17] R. Gonzalez and R. Woods, “Digital Image Processing”, Addison-Wesley Publishing Company, 1992, p 189-190

[18] Tony Lindeber, “Scale Space Theory: A Basic Tool for Analyzing Structures at Different Scales”. 1994

[19] David G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints”, 2004, Chapter 3, Detection of Scale-space extreme. pp 5.

[20] David G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints”, 2004, Chapter 3.2, Local extreme detection. pp 6-7.

[21] "Method and apparatus for identifying scale invariant features in an image and use of same for locating an object in an image", David Lowe's patent for the SIFT algorithm, March 23, 2004

[22] David G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints”, 2004, Chapter 4, Accurate Keypoint Localization. pp 10-12.

[23] David G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints”, 2004, Chapter 5, Orientation Assignment. pp 13-14.

[24] David G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints”, 2004, Chapter 6, The local Image Descriptor. pp 10-12. And Kirchner, Matthew R. "Automatic thresholding of SIFT descriptors." In *Image Processing (ICIP), 2016 IEEE International Conference on*, pp. 291-295. IEEE, 2016.

[25] Johansson, R. S. Westling, G; Bäckström, A.; Flanagan, J. R. (2001). "Eye–hand co-ordination in object manipulation". *Journal of Neuroscience*. **21** (17): 6917–6932.

[26] Morgan Quigley,“Programming Robotics with ROS ” (2015). 219-230

[27] W. S. Newsman (2018), “A Systematic Approach to Learning Robot Programming With ROS”, pp. 103-111.

[28] Patrick Beeson and Barrett Ames, 2015, TRAC-IK: An Open Source Library for Improved Solving of Generic Inverse Kinematics. And Moveit Tutorials Version Kinetic,http://docs.ros.org/kinetic/api/moveit_tutorials/html/doc/trac_ik/trac_ik_tutorial.html

[29] Company: Universal Robot A/S, “UR5 Manual / CB3”, Manipulator Joint Angle Specifications, I-61.

[30] Company: Microsoft, “Kinect Manual”, Kinect Depth data sheet. [Online] Available: www.xbox.com/support

[31] “Robotiq 2 Fingers 85cm – 140cm manual”

[32] Cowlishaw, M. F. Fundamental requirements for picture presentation (PDF). Proc. Society for Information Display. 1985, **26** (2): 101–107.

[33] Agoston, Max K. (2005). Computer Graphics and Geometric Modeling: Implementation and Algorithms. London: Springer. pp. 300–306.

[34] Alvy Ray Smith. Color Gamut Transform Pairs (1978).

[35] J. Flusser: "On the Independence of Rotation Moment Invariants", Pattern Recognition, vol. 33, pp. 1405–1410, 2000.

[36] Open Source from GitHub: https://github.com/ros-industrial/universal_robot

[37] Open Source from GitHub: https://github.com/ros-industrial/robotiq

[38] Open Source from GitHub: https://github.com/wangxian4423/kinect_v2_udrf

Appendix

Minutes

Date and time of meeting:	9/13/2019 16:00~17:00
Number of meeting:	1
Venue of meeting:	GYM
Participants:	Dr Sen Wang
	Ziyu Du, Yuanyue You, Kaicheng Zhang, Zhuocheng Han, Zhanheng Li
Signure:	Sen Wang

Agenda

- Element 1: Self introduction
- Element 2: Demonstration of the previous project
- Element 3: Brief introduction the project objectives

To do:

- Element 1: Learn robot operating system
- Element 2: think about project topics
- Element 3: Read related papers on topics of interest to understand the prospects for the current development of the topic

Meeting record

Date and time of meeting:	9/29 /2019 16:00~17:00
Number of meeting:	2
Venue of meeting:	PG 305
Participants :	Dr Sen Wang
	Ziyu Du, Yuanyue You, Kaicheng Zhang, Zhuocheng Han, Zhanheng Li
Signure:	

Agenda

- Element 1: Project topics selection discussion

To do:

- Element 1: Learn robot operating system
- Element 2: Think about project topics
- Element 3: Read related papers on topics of interest to understand the prospects for the current development of the topic

Meeting record

Date and time of meeting:	10/9 /2019 16:30~17:30
Number of meeting:	3
Venue of meeting:	PG 305
Participants :	Dr Sen Wang
	Ziyu Du, Yuanyue You, Kaicheng Zhang, Zhuocheng Han, Zhanheng Li
Signure:	Sen Wang

Agenda

- Element 1: Project feasibility discussion
- Element 2: Project topics selection discussion
- Element 3: Consult the communication method mechanism in ros

To do:

- Element 1: Learn robot operating system
- Element 2: Learn manipulator Development prospects
- Element 3: Judge whether the difficulty of the project is acceptable
- Element 4: Read related papers on topics of interest to understand the prospects for the current development of the topic

Meeting record

Date and time of meeting:	17/10 /2019 16:30~17:30
Number of meeting:	4
Venue of meeting:	PG 305
Participants :	Dr Sen Wang
	Ziyu Du, Yuanyue You, Kaicheng Zhang, Zhuocheng Han, Zhanheng Li
Signure:	Sen Wang

Agenda

- Element 1: Demonstrate manipulator model in simulator
- Element 2: Consult camera calibration
- Element 3: Consult the communication method mechanism in ros
- Element 4: discuss whether the difficulty of the project is acceptable

To do:

- Element 1: Learn robot operating system
- Element 2: Find Kinect description file in GitHub
- Element 3: Learn manipulator motion planning algorithm
- Element 4: Learn OpenCV recognition method

Meeting record

Date and time of meeting:	1/11 /2019 10:30~11:30
Number of meeting:	5
Venue of meeting:	EM 2.04
Participants :	Dr Sen Wang
	Ziyu Du, Yuanyue You, Kaicheng Zhang, Zhuocheng Han, Zhanheng Li
Signure:	

Agenda

- Element 1: Demonstrate camera calibration results
- Element 2: Discuss presentation considerations
- Element 3: Consult the moveit ros
- Element 4: discuss whether the difficulty of the project is acceptable

To do:

- Element 1: Learn robot operating system
- Element 2: Read sift related paper
- Element 3: Build whole vision based manipulator system
- Element 4: Learn OpenCV recognition method

Meeting record

Date and time of meeting:	14/11 /2019 10:30~11:30
Number of meeting:	6
Venue of meeting:	EM 2.04
Participants :	Dr Sen Wang
	Ziyu Du, Yuanyue You, Kaicheng Zhang, Zhuocheng Han, Zhanheng Li
Signure:	Sen Wang

Agenda

- Element 1: Demonstrate part of grasping system model
- Element 2: Discuss presentation considerations
- Element 3: Consult OpenCV recognition method
- Element 4: Consult manipulator motion planning algorithm

To do:

- Element 1: try to complete OpenCV recognition
- Element 2: Read sift related paper
- Element 3: build a moveit manipulator package and try to understand parameters. in it
- Element 4: continuously build vision based manipulator grasping system

Meeting record

Date and time of meeting:	22/11 /2019 10:30~11:30
Number of meeting:	7
Venue of meeting:	EM 2.04
Participants :	Dr Sen Wang
	Ziyu Du, Yuanyue You, Kaicheng Zhang, Zhuocheng Han, Zhanheng Li
Signure:	Sen Wang

Agenda

- Element 1: Demonstrate part of grasping system model
- Element 2: demo moveit package of manipulator and consult some moveit. problem
- Element 3: demo OpenCV recognition method
- Element 4: Consult manipulator inverse kinematics problems

To do:

- Element 1: try to complete OpenCV recognition and use find-object package. recognize target
- Element 2: Read sift related paper and understand how to set appropriate。parameters
- Element 3: perfect moveit function of manipulator and make the end effect of robot arm can move to a target position
- Element 4: continuously build vision-based manipulator grasping system

Meeting record

Date and time of meeting:	10/12 /2019 10:30~11:30
Number of meeting:	8
Venue of meeting:	EM 2.04
Participants :	Dr Sen Wang
	Ziyu Du, Yuanyue You, Kaicheng Zhang, Zhuocheng Han, Zhanheng Li
Signure:	Sen Wang

Agenda

- Element 1: Demonstrate opencv recognition method
- Element 2: demo controlling manipulator
- Element 3: Consult manipulator inverse kinematics problems
- Element 4: Consult sift parameters problems

To do:

- Element 1: try to complete OpenCV recognition and use find-object package. recognize target
- Element 2: perfect sift parameters
- Element 3: try to write a publisher to publish topic and a subscriber to subscribe a topic
- Element 4: continuously build vision-based manipulator grasping system

Meeting record

Date and time of meeting:	26/2 /2020 10:30~11:30
Number of meeting:	9
Venue of meeting:	EM 2.04
Participants :	Dr Sen Wang
	Ziyu Du, Yuanyue You, Kaicheng Zhang, Zhuocheng Han, Zhanheng Li
Signure:	Sen Wang

Agenda

- Element 1: demo : recognition by find-object package, sift algorithm
- Element 2: demo : the end effect of manipulator can move to pre-grasp position
- Element 3: demo : OpenCV recognition method
- Element 4: demo: manipulator and gripper pre-set position
- Element 5: discuss: accuracy of sift and OpenCV

To do:

- Element 1: do OpenCV pixel coord to world coor calculation
- Element 2: campare two methods
- Element 3: debug workspace error
- Element 4: continuously build vision-based manipulator grasping system

Meeting record

Date and time of meeting:	10/3 /2020 12:00~1:00
Number of meeting:	10
Venue of meeting:	video meeting
Participants :	Dr Sen Wang
	Ziyu Du, Yuanyue You, Kaicheng Zhang, Zhuocheng Han, Zhanheng Li
Signure:	Sen Wang

Agenda

- Element 1: demo : bision-based manipulator grasping system
- Element 2: discuss : matrix conversion problems

To do:

- Element 1: complete OpenCV pixel coord to world coor calculation
- Element 2: campare two methods
- Element 3: debug errors
- Element 4: continuously build vision-based manipulator grasping system

Meeting record

Date and time of meeting:	17/4 /2020 10:30~11:30
Number of meeting:	11
Venue of meeting:	video meeting
Participants :	Dr Sen Wang
	Ziyu Du, Yuanyue You, Kaicheng Zhang, Zhuocheng Han, Zhanheng Li
Signure:	Sen Wang

Agenda

- Element 1: demo : bision-based manipulator grasping system
- Element 2: discuss : thesis problems

To do:

- Element 1: perfect the grasping system
- Element 2: complete project thesis