

國立臺灣大學電機資訊學院電機工程學研究所



碩士論文

Graduate Institute of Electrical Engineering
College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

基於視覺伺服控制之機器人輸送帶追蹤技術及 3D 多異種物件
抓取技術

Visual Servo Control Based Robotic Conveyor Tracking and
Dynamic 3D Heterogeneous Object Fetching

廖俊豪

Chun-Hao Liao

指導教授：羅仁權 博士

Advisor: Ren C. Luo, Ph.D.

中華民國 106 年 7 月

July 2017

誌謝



回想兩年前，還在人生道路上迷惘與徬徨，如今卻已經碩士班畢業了。這兩年的研究生活對我來說真的別具意義，大學時總是被動地接受老師給予的知識，而研究所則要自己主動探索知識、發現問題、面對問題並且解決問題。謝謝這段時間以來家人的支持與鼓勵，因為研究的關係偶爾才能回家一次，但每次回家都是充電，謝謝我的家人總是鼓勵並支持我，讓我能無後顧之憂專注在研究上。很感謝指導老師羅仁權教授，實驗室一直都是以工業型機器人為主，在提出機械手臂的動態追蹤夾取的研究構想上，老師親自提出許多問題，期許我朝問題的方向去發展，這給我往後的研究相當大的鼓勵與信心。而之後老師也傾全力的支持我的研究，不論是提供資源或是在研究上給予新的想法都對我受益許多。

在國立臺灣大學智慧機器人及自動化國際研究中心 (NTU- iCeIRA) 兩年的研究生活中，我要感謝鏡文、瑋隆、東榕、繼棠、金成、昕昞、旭佳、禮聰、志遠、獻章等博班學長姐。尤其是鏡文、瑋隆學長，當我在比賽、研究或人生規劃上有問題時，學長們不僅不厭其煩的給我指導，並且也教我如何自己摸索出問題的答案。還要感謝碩班學長銘駿、建安、士紘、煒森、文謙、金博、建偉、冠志、柏宏、榮育，從優秀的學長姐們身上我學到很多，也常把他們當做努力的榜樣，期許自己也能跟學長姐們一樣厲害。更不能忘記同居一起努力奮鬥的伙伴莉彤、長鈞、李晟、晴岡、靖霖、昱佑、達方、凱鈞、仲凱、孟勳、柏凱，和你們一起窩在實驗室，不論是做研究、忙比賽、耍廢還是吃宵夜，都是我碩班生涯最快樂的事情之一。謝謝積極認真又貼心的學弟妹們，培淳、石崑、武昱、嵩詠、智堅、威辰、錦賢、育榕、育澤、名彥、展嘉、何鑫、王昊、曾旻，幫忙大大小小的事務，一起打球運動。也要感謝默默在背後幫我們完成許多瑣事的莉莉(Lily)、雯雅(Tracy)、煜倫(Dornin)、姿伶(Amy)、芳嫻(Helen)、佩芸(Winnie)等助理們。最後特別感謝三年前獨立完成 iCeIRA arm 的昀軒學長，沒有學長之前的努力與貢獻，就沒有這麼好的實驗平台能夠完成自己想做的研究，謝謝學長。

廖俊豪 謹誌

民國一百零六年七月

doi:10.6342/NTU201703382

中文摘要



目前工廠自動化發展的瓶頸是執行任務間，人與機器的互動模式。機器人為了可以在生產線上快速地進行辨識和抓取，像人一樣具備高準確率之外，也要能夠感知外在環境發生的變化。在現行的機器人研究領域，物件追蹤技術在生產線裝配上已經屬於機械手臂的基本技術。傳統而言，機器人執行任務都是步驟進行，一旦第一個步驟失敗，將會導致接續的子步驟，難以進行。

現況而言，要在工業上解決這個問題，多半是依靠視覺系統的輔助為主。這是其中一種機器人的感知系統，機器人可以藉著他們的輔助做好事前準備，能更完美的完成任務。然後，每當無預期的事件發生時，傳統固定形式的步驟，由於不具備動態更新動作，不僅造成任務突然終止，也容易對環境造成不良的影響。即使依靠了視覺輔助，也僅能解決部分的突發狀況。

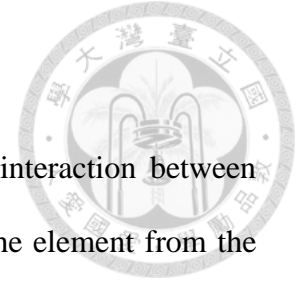
要解決這個問題的方法，一樣還是要依靠機器人視覺。在傳統情境中，輸送帶上的物體是靜止不動，機器人會根據命令進行物件抓取。當目標物開始在生產線上移動，任務的複雜性也隨之提升。這時使用視覺回授系統會是絕佳的解決方法。

因此本研究主題，提供了一個視覺辨識抓取兼追蹤系統，每次抓取姿態都是由回授系統來決定。這個系統架構分兩塊，各自擁有核心的演算法來進行物件追蹤。因為受限於環境的條件，整體實驗操作情形會有細節描述。此外對於追蹤時的抓取姿態，也有改良演算法去記錄補償值，有利於下一次的追蹤。

關鍵字：機器人視覺系統、物件抓取、追蹤系統、工廠自動化

doi:10.6342/NTU201703382

ABSTRACT



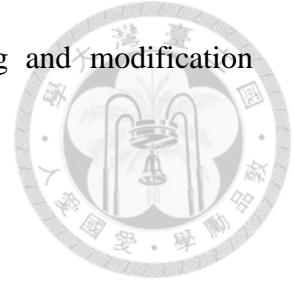
One of the bottlenecks for manufacturing automation is the interaction between robot and humans among tasks. Robots are not able to recognize the element from the assembly line quickly and accurately just like human operators do. Besides, in recent robotic field, conveyor tracking is one of fundamental function in the robot manipulator. Once this very first step fails in the production line, the latter subsequent operations are hardly to complete.

Currently, the manufacturers solve this problem by using the vision in the environment. The robot arm now can perform the task and manipulate the work based on the precondition. Afterwards, something that may occur will be regarded as a kind of unpredictable circumstance among the task. Due to traditional static process, the condition may not only cancel the work but also have bad impacts on the environment. The problem is also about vision.

In order to solve the dilemma, the practicable way is also based on robot vision system. In traditional scenario, the object keep stable pose on the assembly line. The manipulator follows commands to grasp the object. While the target is moving on the production line, the task becomes sophisticated problems. Thus, a distinct grasping method under visual control system is definitely one of the essential solutions.

In this thesis, we propose a tracking strategy on moving objects for a robot arm object fetching system combined with distinct recognition algorithm. In addition, the grasping pose of robot arm is correct by visual feedback system. The system is separated into two parts: eye to hand and eye in hand, and discussed in detail. Each part owns its core algorithm to complete industrial tracking and fetching tasks. Because of limitation from the environment, the working conditions will also be illustrated.

Grasping pose for each type of element is adjusted by tracking and modification algorithms.



Keywords: robotic vision system, object fetching, tracking system, factory automation

CONTENTS



口試委員會審定書

誌謝i

中文摘要 ii

ABSTRACT iii

CONTENTSv

LIST OF FIGURES viii

LIST OF TABLESxi

Chapter 1 Introduction 1

1.1 History 1

1.1.1 Traditional industrial robot arms 1

1.1.2 Lightweight payload robot arms 2

1.2 Industrial Applications 4

1.2.1 Object fetching 5

1.2.2 Assembly 7

1.3 Challenges 7

1.3.1 Object recognition 7

1.3.2 Robot vision system 8

1.4 Thesis Structure 10

Chapter 2 Scenario 11

2.1 Experimental Setup 11

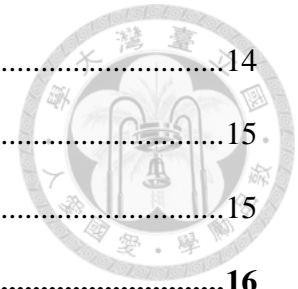
2.1.1 Scene 11

2.1.2 Faced problems 12

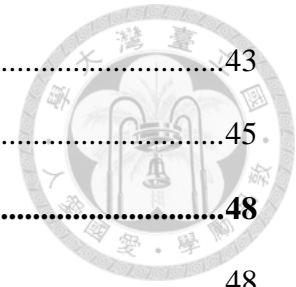
2.2 Procedures 12

doi:10.6342/NTU201703382

2.3	Preconditions	14
2.3.1	Structured environment	15
2.3.2	Objects description	15
Chapter 3	System Architecture	16
3.1	Generalized Robot Fetching Architecture	16
3.2	Specialized Robot Fetching Architecture	17
3.3	Modified Robot Fetching Architecture	19
Chapter 4	Manipulator	21
4.1	Mechanism.....	21
4.1.1	D-H parameters.....	22
4.1.2	Transmission and actuator	25
4.1.3	Gripper.....	27
4.2	Control Architecture	30
4.3	Software Architecture	31
4.3.1	Motivation	32
4.3.2	Hardware Structure.....	32
4.3.3	Basic Utility.....	33
4.3.4	Application Layer.....	33
4.3.5	Timer.....	34
4.4	Manipulator Functionalities.....	35
4.4.1	Intuitive teaching by touching	35
4.4.2	Online trajectory generation	37
Chapter 5	Object Recognition.....	41
5.1	Point Cloud Library	41
5.2	Database Generation.....	42



5.3	Kinect Calibration.....	43
5.4	Object type and pose recognition	45
Chapter 6	Object Tracking Strategy	48
6.1	Problem Statement.....	48
6.2	Tracking Algorithm.....	49
6.2.1	Image segmentation	50
6.2.2	Position localization	51
6.2.3	Position calibration	52
6.3	Modified Grasping.....	52
6.3.1	Moving calibration	53
6.3.2	Pose modification	54
6.3.3	SVM database.....	56
Chapter 7	Experimental Results and Discussion.....	66
7.1	Object Recognition and Fetching	66
7.2	Object Tracking	68
7.3	Modified Grasping.....	72
7.3.1	Object pose modification.....	72
7.3.2	Classifier results	73
Chapter 8	Conclusions, Contributions and Future Works.....	77
REFERENCE	78
VITA	84



LIST OF FIGURES



Fig. 1.1 FANUC R1000 series robots	2
Fig. 1.2 Lightweight payload robot arms	3
Fig. 1.3 The PRob 1R collaborative robot	4
Fig. 1.4 The Kuka KR Agilus is playing table tennis	4
Fig. 1.5 Arm fetch an object	6
Fig. 1.6 Robot assembly	6
Fig. 1.7 Intuitive teaching by touching block diagram	9
Fig. 2.1 An industrial task outlining the pickup of moving objects.....	11
Fig. 2.2 The light sensor of the conveyor	12
Fig. 2.3 Eye to hand system (Microsoft Kinect sensor)	14
Fig. 2.4 Eye in hand system (webcam camera)	14
Fig. 3.1 Generalized robot fetching architecture	16
Fig. 3.2 Flow chart of specialized robot fetching system architecture.	17
Fig. 3.3 System structure for modified robot fetching.	19
Fig. 4.1 iCeIRA arm one in iCeIRA lab.....	21
Fig. 4.2 The definition of four parameters.....	23
Fig. 4.3 Link frames assignment of iCeIRA arm one	24
Fig. 4.4 Joint structure of robot arm	26
Fig. 4.5 The figure of motor, driver and gear box	26
Fig. 4.6 Robotiq 3-finger gripper	27
Fig. 4.7 Different type of grip.....	27
Fig. 4.8 Dimension and workspace of gripper	28
Fig. 4.9 Four operation modes of the gripper	29

Fig. 4.10 Control architecture in robot control system.....	30
Fig. 4.11 Software architecture.....	31
Fig. 4.12 The input and output of an OTG algorithm.....	39
Fig. 5.1 Point Cloud illustration	41
Fig. 5.2 The camera distortion on the projection plane	43
Fig. 5.3 The PCL translates the Kinect coordinate to reference coordinate	45
Fig. 5.4 Three types of objects in the pick and place task.....	46
Fig. 6.1 The position of two sensors in experiment.....	48
Fig. 6.2 The tracking diagram	49
Fig. 6.3 Line description in polar coordinate.....	53
Fig. 6.4 System Structure for modified grasping.	55
Fig. 6.5 A block diagram of a negative feedback control system.....	56
Fig. 6.6 Maxon motor control.....	57
Fig. 6.7 Intelligent motion control platform	57
Fig. 6.8 The artificial calibration illustration.....	59
Fig. 6.9 Two dimension example in SVM analysis	62
Fig. 6.10 The correct pose of the element.	64
Fig. 6.11 The skewed pose of the element.....	65
Fig. 7.1 Experimental Scenario	66
Fig. 7.2 The working area of object recognition system	67
Fig. 7.3 The ROI of each object	67
Fig. 7.4 Object recognition results.....	67
Fig. 7.5 Object fetching results.....	68
Fig. 7.6 Simulation of tracking an object	69
Fig. 7.7 The process of eye to hand tracking.....	70

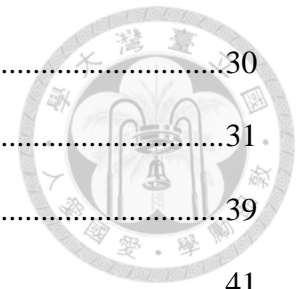
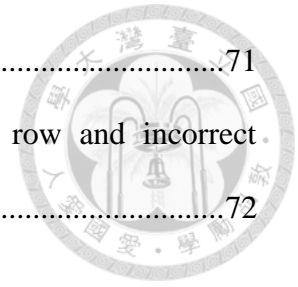


Fig. 7.8 The process of eye in hand tracking71

Fig. 7.9 The conveyor edge detection. Correct grasping in top row and incorrect grasping in bottom row.72



LIST OF TABLES



Table 4-1 spec of the iCeiRA arm one.....	22
Table 4-2 D-H table of iCeiRA arm one.....	25
Table 4-3 Transmissions and actuators	26
Table 4-4 spec of gripper	28
Table 7-1 Single-class Lib-SVM recognition result	73
Table 7-2 Intra-class statistics.....	76
Table 7-3 Inter-class statistics.....	76

Chapter 1 Introduction



1.1 History

For many years, the robotic technology has been widely applied to our industry in order to enhance the performance of production in factories. Usually, when the industrial elements have been processed, the associated step may be doing further procedures such as cutting, gluing, welding, coating, deburring, and etc. Among different process, the element is held by the robot manipulator to move from this stage to the next one.

1.1.1 Traditional industrial robot arms

Traditional industrial robot arms feature high payload, high speed, high precision, and high stiffness. A typical example of FANUC robot arm is shown in Fig. 1.1. They are usually large in size for a large workspace and actuated by hydraulic pump for high payload and stiffness. Most of them are 6 degree of freedoms or less depending on the target application. Those robots are targeting on tasks that require much strength high precision and tasks they are boring and repetitive or dangerous such as material handling, welding, automotive manufacturing, etc. Due to their strength, the robot arm must be fenced when they are working and no humans are allowed to enter the workspace. This excludes the possibility of human robot co-existence and human robot collaboration. Although these robots hardly cooperate with humans directly, the programming is often based on teaching by touching technique. With more and more application combined with vision system in small size of robot manipulators, there are some research topics to build up vision-based system in the robot assembly lines [1]. The function in the big robot is no longer restricted in the industrial application.



Fig. 1.1 FANUC R1000 series robots

1.1.2 Lightweight payload robot arms

On the other hand, some researchers and companies tried to lift robot to a higher level by taking the human-robot interaction (HRI) or collaboration (HRC) into consideration. Under the concept of HRI and HRC, the robot arm should be relatively smaller than traditional industrial robot arms. That is to say, some design concepts will follow the request to develop the skill under the circumstances of human-robot coexistence. For example: (a) ABB IRB 1200 is one of lightweight robot arm. Its sleek, smooth surfaces and enclosed design feature all wiring and air routed through the inside of the robot, from very close to the wrist flange all the way to the foot. This further enhances its compactness, and makes the new robot easy to maintain and clean, with no risk of dirt or dust collecting on the cables. (b) Barrett Technology WAM arm is controlled by several steel elastic strings and it is relatively safer than conventional motor driven robot. All the robots mentioned above are listed in Fig. 1.2.



(a)



(b)

Fig. 1.2 Lightweight payload robot arms

One of the most important requirements for collaborative robot is safety. We don't want the robot hurt the human that cooperate with her. As a result, they are all capable of sensitive collision detection. The PRob 1R collaborative robot, in Fig. 1.3, was developed to make customers' lives easier. In fact, when most people think about robotics, they are usually afraid of programming complex routines with a non-intuitive platform. With the help of lightweight design, a modular end effector with soft material makes it safe for humans. There is a switch made through a patented-protected interface that allows a mechanical and electrical connect-disconnect operation to take place. The fingertips of the robot gripper can be adapted for specific application and are easy to switch once the robot is in operation. Another important feature for human robot collaboration is that the robot should be able to interact with human by some gestures. For example, a famous file shows man play table tennis with KUKA KR Aglius arm [2]. The arm hitting the ball is shown in Fig. 1.4. Due to its high levels of precision at fast speeds, the world No 1 company shows astonishing game to the society that this arm with incredible agility will show reaction after every human gesture. This kind of application verifies that the high precision in the lightweight robot arm can finish the work in a common but fascinating way.

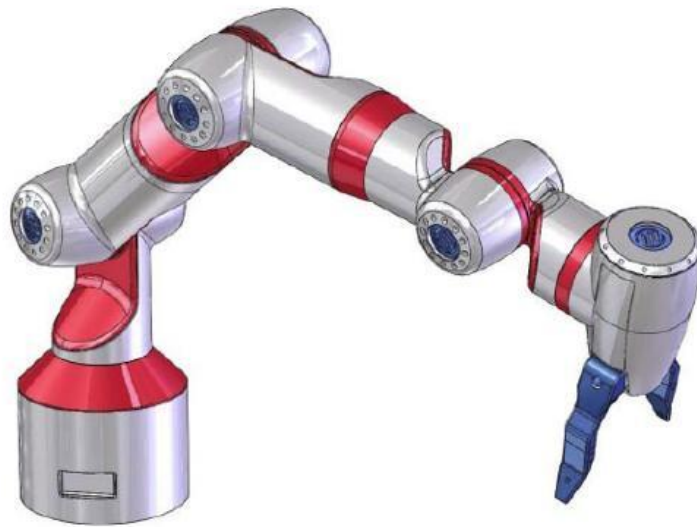


Fig. 1.3 The PRob 1R collaborative robot

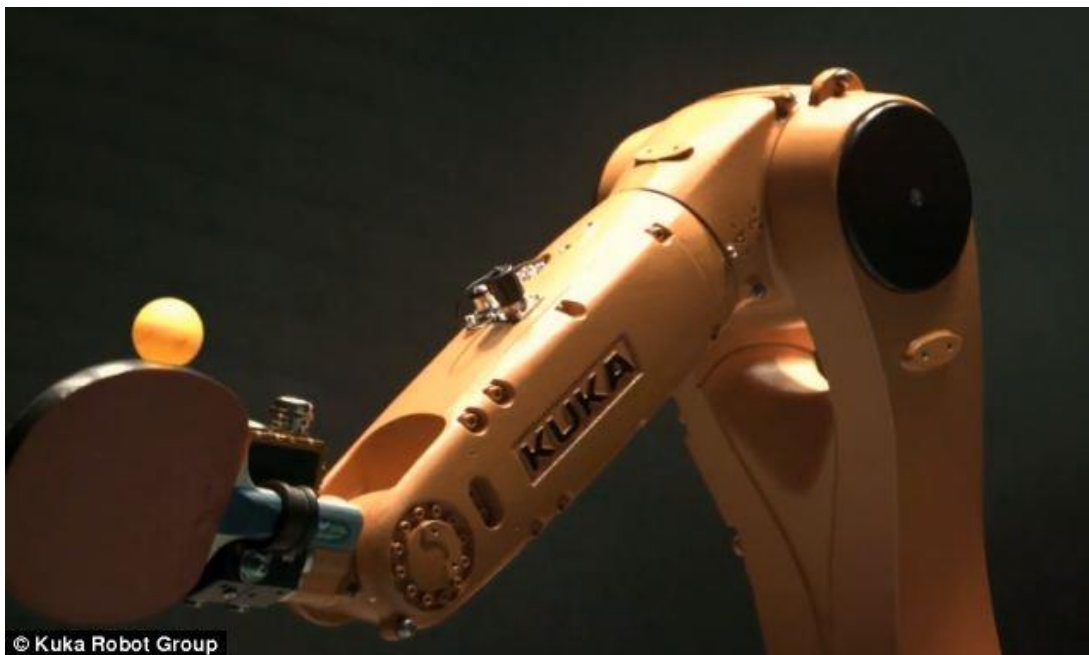
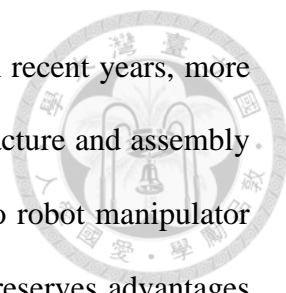


Fig. 1.4 The KUKA KR Agilus is playing table tennis

1.2 Industrial Applications

For industrial robotic applications utilizing a robot arm, the most fundamental functionality required is that the arm should be able to move from the current pose to the target pose so that it picks the machined objects to place on the platform.

Looking back on the industrial production line, factory automation equipped with



various industrial mechanisms has already become a mainstream. In recent years, more and more applications have been combined with mechanical manufacture and assembly under the same production line. The automatic part usually refers to robot manipulator operation with intelligent skills. Actually, the intelligent machine preserves advantages to be strong and feasible. For example, the tasks in the production line often contain solid and simple actions. If humans have unlimited time and energy, the work will remain the same. However, humans have to rest periodically to stay efficient. However, robots can retain its performance at a certain level for much longer. Main tasks for machines on assembly lines include processing, assembling, packaging and distributing products. Furthermore, robots or machines enabled to grasp huge workpieces have not yet been invented. Based on diverse applications, operations are designed to solve problems in the production line.

1.2.1 Object fetching

To enhance the productivity in the industrial production line, robots will replace human in the labor intensive works which usually are repeated action. The advantages are to decrease the cost and to make sure the quality of products during the process. It seems that the robot manipulator is indispensable to help humans complete the challenges in the production line. Raising a new method is a trend to establish the capability of robot. The most common work is about object fetching, which is also called pick and place task.

The technique for grasping objects such as industrial products should be developed and teach robot manipulators on the assembly lines. First, machine tools manufacture various industrial products respectively. When those things are produced, the way to put on the conveyor is to grab them with proper command to robot arms. Actually, element gripping happens among different working processes. In [3] and [4], this is the common

method to pick component from one place to the target position. This can easily be done by repeating mechanical commands. To accomplish more complicated tasks, element information is essential for robots. Sensors play an important role in the scheme of targets on the assembly [5]. Sensory data from vision sensors usually contain the position and orientation of objects on the conveyor. [6] and [7] provide a technique to tell robot where and when to grab elements under designation.

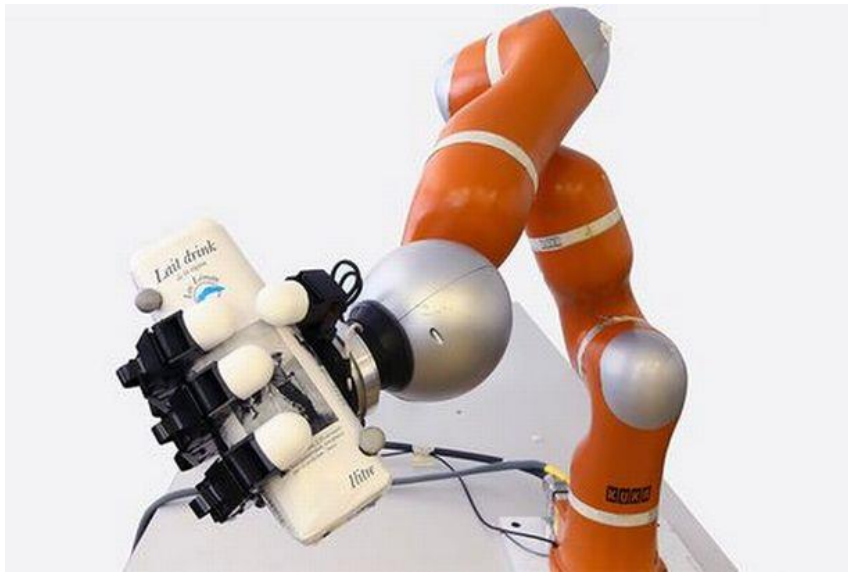


Fig. 1.5 Arm fetch an object

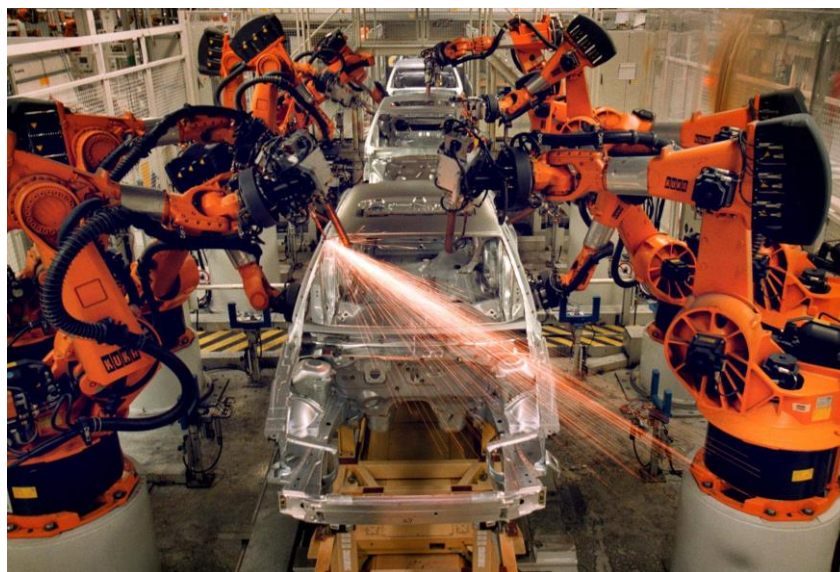
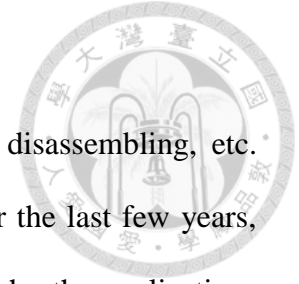


Fig. 1.6 Robot assembly



1.2.2 Assembly

Assembly operations include: fixing, press-fitting, inserting, disassembling, etc. This category of robotic applications seems to have decreased over the last few years, even while other robotic applications have increased. The reason why the applications are diversified is because of the introduction of different technologies such as force torque sensors and tactile sensors that gives more sensations to the robot.

However, the assembly is still a common application in industrial automation system. Actually, the transmission part would be a fixed conveyor to deliver the element. At this stage, the task for robot manipulators is simple to move from one place to pick and to target point to place periodically. Besides, without any device to check the position and orientation, there are some challenges happening during the task.

1.3 Challenges

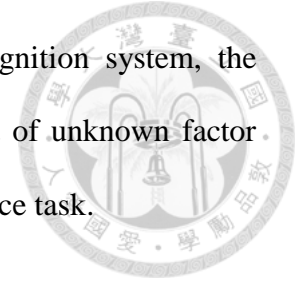
There is a significant bottleneck in the development of industrial automation. The challenge is robot vision system. Due to the recognition performance in the factory, the robot has difficulty finding and picking the elements on the conveyor precisely.

When the target starts to change its movement on the production line, the challenge turns out to be more complicated without any feedback control system. Undoubtedly, a visual system helps to track the moving target and transmits new data to the robot arm. As a result, a new grasping technique under visual control system is definitely required.

1.3.1 Object recognition

An object recognition system finds objects in the real world from an image of the world, using object models. This task is surprisingly difficult if we want to implement better effort in the robot. Humans perform object recognition effortlessly and instantaneously. However, the algorithmic has huge time-consumption in clustering the

region and recognizing the target. But by means of object recognition system, the manipulator has an eye to understand the surrounding where a lot of unknown factor exists. Therefore, this system plays an important role in pick and place task.



1.3.2 Robot vision system

Visual camera is a common piece of equipment in the robotic picking tasks. The robot vision system is considered as assistant to finish work. The 2D data is derived from webcam and transformed into poses for robot arms [8]. Some image processing algorithms have high impact on object recognition, increasing the success rate of grabbing [9]. Moreover, Kinect sensors are famous for its depth information, which assist robot end-effectors in grasping. [10] and [11] show the advantages of depth data to enhance entire process of conveyor tracking. Compared with a laser scanner, a tilting laser range finder, the 3D geometry sensors in [12] would be more straightforward to describe element situation from elements to robot arms. As a result, cameras are one of the key points under pick and place tasks.

However, there are a lot of unexpected conditions to consider in the assembly lines and these problems need to be solved. In this condition, the unexpected part is moving conveyor. Sensor integration brings about a comprehensive conception on robot grasping. Equipped more sensors can not only deal with complex environment but also decreases the error rate of a task [13]. There are two ways to describe the relationship between arms and cameras. One is camera to hand and the other is eye-in-hand. Camera to hand is common to send background information to control center. Eye-in-hand usually transmits object state to computer to revise the position and orientation of targets immediately if problems happen in the environment. Moreover, the movement of robot manipulators is not always stable. To strengthen the function in perception system, the condition should be recorded so that all motion will remain more fluent next time.

doi:10.6342/NTU201703382

Besides, with the help of communication between robot and PC, the development of robot can be implemented by programming. With additional and various sensors such as vision, force, light, and current, the robot arm can not only fully reach to the target point but also integrate with sensor data in a flexible way. However, in the developer part, it takes tediously long time to integrate different system signal to deal with numerous changes happening to the assembly lines. Probably the work is only for customized design. To match more robots in the market, if the program is not suitable for more device or more sensors, the process for robot manipulator may change a lot to even to rewrite the whole control system. Therefore, visual feedback control is a relative convenient system to cope with the problems suddenly happening with the help of content from vision sensors. A famous technique, intuitive teaching by touching, is often used as visual feedback system to complete robot grasping. A block diagram is shown in Fig. 1.7.

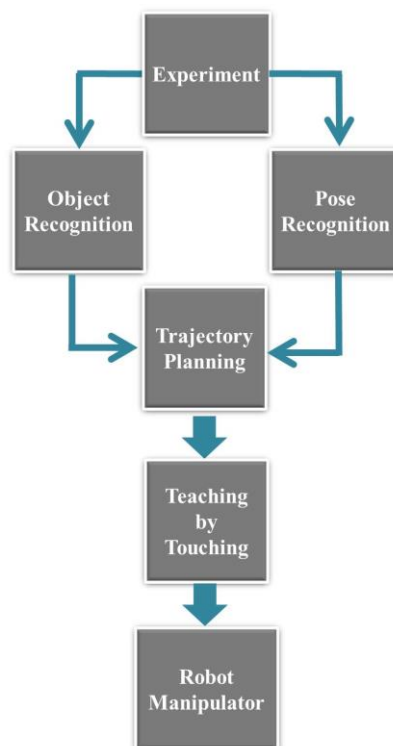
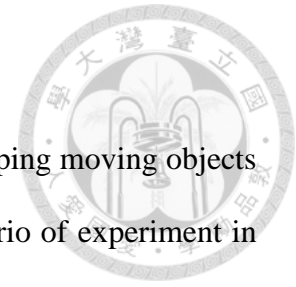


Fig. 1.7 Intuitive teaching by touching block diagram

1.4 Thesis Structure

In this thesis, we introduce the idea of tracking strategy to grasping moving objects on the assembly lines. In Chapter 2, we briefly introduce the scenario of experiment in the thesis, including the problems to solve, the functionalities to build and the procedures to implement. There are two kinds of situation in the industrial elements. In Chapter 3, there are three main architectures to illustrate the process implemented in this research. The second one and third one is about tracking method. After pointing out the thesis structure, we start to introduce each method in the following chapters. In Chapter 4, the manipulator will be illustrated in solid part and programming. Because the robot is the center of this technique, the fundamental functionalities will be shown in this chapter. In Chapter 5, we will focus on how the robot knows to grasp the object by recognition system. The main technique, tracking strategy, will be introduced in Chapter 6. The process to complete the pick and place task is written in details. In Chapter 7, the experimental results will be shown and discuss them in comparison with theoretically prediction. Last of all, we conclude whole thesis with future works in Chapter 8.





Chapter 2 Scenario

2.1 Experimental Setup

The main goal in this research topic is to pick up a moving object from the conveyor to the platform. The associated factors to complete the task will be fully illustrated in the following subsections.

2.1.1 Scene

The scene of our experiment is illustrated in Fig. 2.1. Objects to be picked up are transmitted by a conveyor. The object is moving among whole process. The manipulator is a 7-DoFs robot arm which is designed and assembled in our lab. There is a 3 finger gripper in the tip point to grasp objects on the moving conveyor. The visual sensors are able to sense any difference happening in the environment. In this research, the system is sensor-integrated with two vision sensors including Microsoft Kinect and webcam camera.

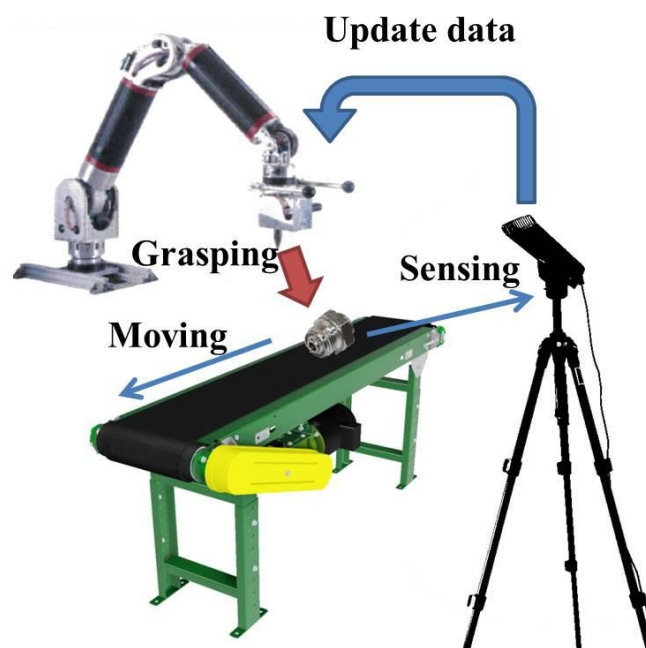
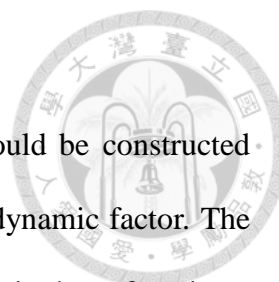


Fig. 2.1 An industrial task outlining the pickup of moving objects



2.1.2 Faced problems

For a proper robot assembly, some fundamental functions should be constructed including manipulator control, object recognition and fetching and dynamic factor. The whole process to list will be introduced in Chapter 4. When the manipulator functions, there are some problems robot must handle. In factories, most of the problems are the instant position and orientation of object. In object recognition system, the lighting condition may also affect the color or intensity feature of the object. When this basic factor hasn't been solved, the robot is unable to hold an object, let alone grasp moving one.

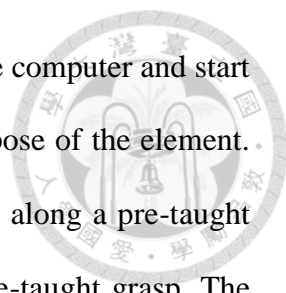
2.2 Procedures

The whole procedures can be separated into two parts: static and dynamic.



Fig. 2.2 The light sensor of the conveyor

In static part, the industrial elements are put on the conveyor and being transmitted to a certain region where the light sensor is mounted. The light sensor in Fig. 2.2 will be trigger and stop the conveyor so that the part will stop and wait for subsequent



operations. At the meantime, the light sensor will send a signal to the computer and start the object recognition algorithm to identify the type as well as the pose of the element. After the success of the object recognition, the manipulator will go along a pre-taught trajectory to the ready pose and then pick up the object using a pre-taught grasp. The manipulator will then start the operations for the object such as assembling, polishing, painting, etc. After the current task accomplished, the manipulator will send a signal to the conveyor and transmit the next part for the next operation. The cycle then repeats itself.

When the conveyor starts to move, there are several dynamic factors on the conveyor. The vision sensors are used as monitors to sense any variation in the experiment. In this thesis, two ways to describe the relationship between arms and cameras will be illustrated. The first way is eye to hand, shown in Fig. 2.3. Robot eye stands at a relative position to the robot hand. This means that the camera is set near by the robot manipulator and its distance remains solid. When the conveyor start to transmit the objects, the camera will check the instant position and recognize the object at the same time. After receiving new data from vision, the end-effector will run to the target to pick up the object. The other one is eye in hand, shown in Fig. 2.4. Literally, on the tooltip of manipulator, there is an eye on it. This eye helps to track the object immediately. However, the view may include partial body of object. To calculate the area of object in the camera frame, it is important to segment the 640 x 480 view. Only a small region is interested in the procedure. The robot manipulator will track the object three times and grasp it before it falls down to the ground. Therefore, the dynamic factors can be solved by two tracking skill. The vision sensor used in eye to hand is Microsoft Kinect, and the other one in eye in hand is webcam camera.



Fig. 2.3 Eye to hand system (Microsoft Kinect sensor)

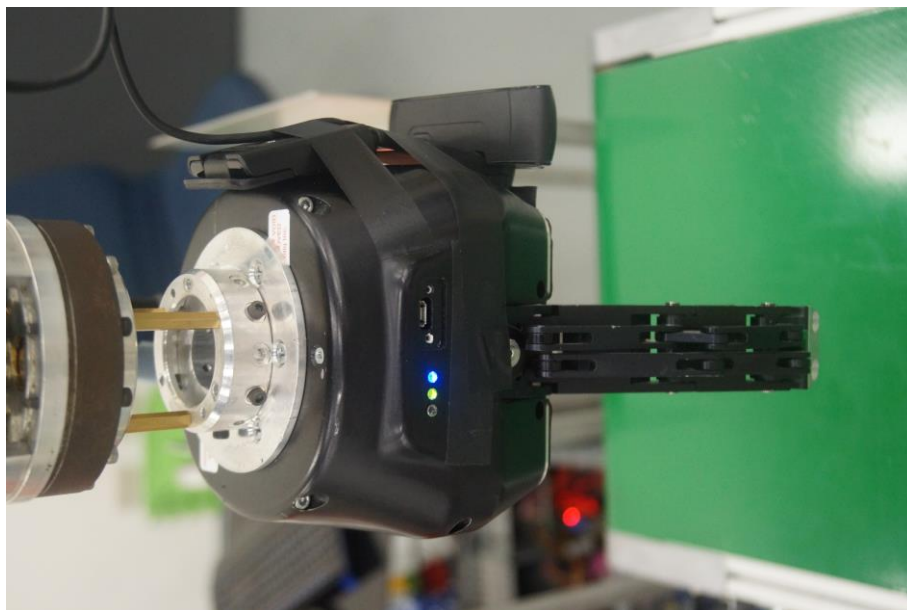


Fig. 2.4 Eye in hand system (webcam camera)

2.3 Preconditions

In industrial scenarios, the environment is rather controllable compared with household scenarios. As a result, with some assumptions properly made, we can greatly increase the efficiency of the system. Furthermore, we can setup the environment to meet the precondition of some algorithms, in our case the object recognition algorithms.

so that those powerful algorithms can be adapted to meet our requirement of the system such as high reliability, low time consumption, and so on.



2.3.1 Structured environment

In industrial scenario, the environment is controllable. We can build fences around the robot which prohibits humans from going inside. The environment can be fixed once the system has been deployed. As a result, if the robot can work in a well-controlled and structured environment, which should not be a big problem setting up such environment in industrial scenarios, the robot will not have to take online motion planning and obstacle avoidance into account. The robot can simply follow a pre-programmed and time-optimal trajectory to go to the target pose and perform its tasks swiftly and safely.

2.3.2 Objects description

We assume that objects or components that may occur in the scenario are all rigid bodies. They would not deform or break on account of external forces. Following this assumption, model-based object recognition algorithms can be applied. The states of the object can be described by its type and the 6-DoFs rigid transform in the 3D Cartesian space. The internal states such as the deformation can be neglected under the rigid body assumption. Furthermore, the feasible grasp for picking up the object stably can be described easily. Other factors such as the exerted force of the gripper, the damage of the object and the deformation of the object are not taken into consideration. Only the grasp point is required for describing the proper grasp, which is simply the 6-DoFs pose of the end-effector in the object's coordinate.

Chapter 3 System Architecture

3.1 Generalized Robot Fetching Architecture



The generalized system architecture and the required su0modules for a robot fetching system are shown in Fig. 3.1. The environment is sensed by 3D sensors that describe the robot's workspace with the color, the geometry, or both. The sensed environment information is then transmitted to the object recognition module which recognizes the target objects in the workspace. The output result is the pose and the type of the target objects in the workspace. Since the type of the objects in the scene is known, their models can be retrieved from the database for grasp planning. The resulted grasp contains the pose of the end-effector and the configuration of the end-effector. The former one should go through a motion planner which generates a collision-free trajectory for robot arm execution, while the latter one will be sent to the end-effector, which could be a gripper, for actually holding the object.

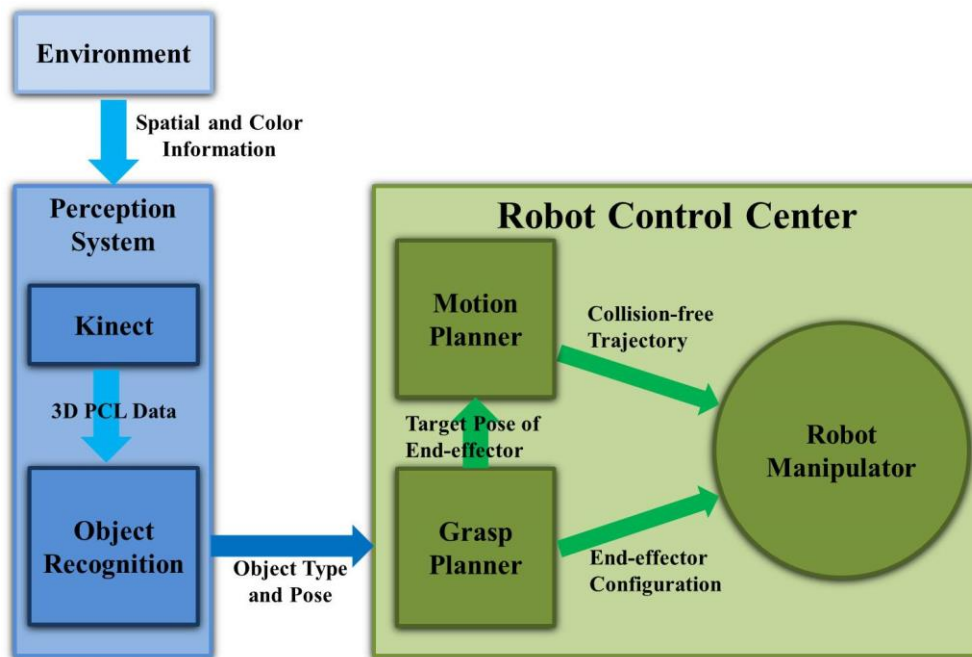


Fig. 3.1 Generalized robot fetching architecture

This architecture, though dealing with general situation, suffers from high computational complexity in the online grasp planning and motion planning. In an industrial scenario, the efficiency, which is closely related to the cost and the yield rate, is one of the major concerns. We not only require the robot to correctly complete the tasks but also need the tasks to be done quickly. As a result, in this research, we based on the assumptions made in the subsection 2.3 and substitute the online grasp planner and the motion planner modules with the operation database and trajectory interpolator respectively.

3.2 Specialized Robot Fetching Architecture

The main objective in this research is to build up a sensor-integrated system that can grab moving objects with pose modification in the assembly lines. The system architecture is shown in Fig. 3.2.

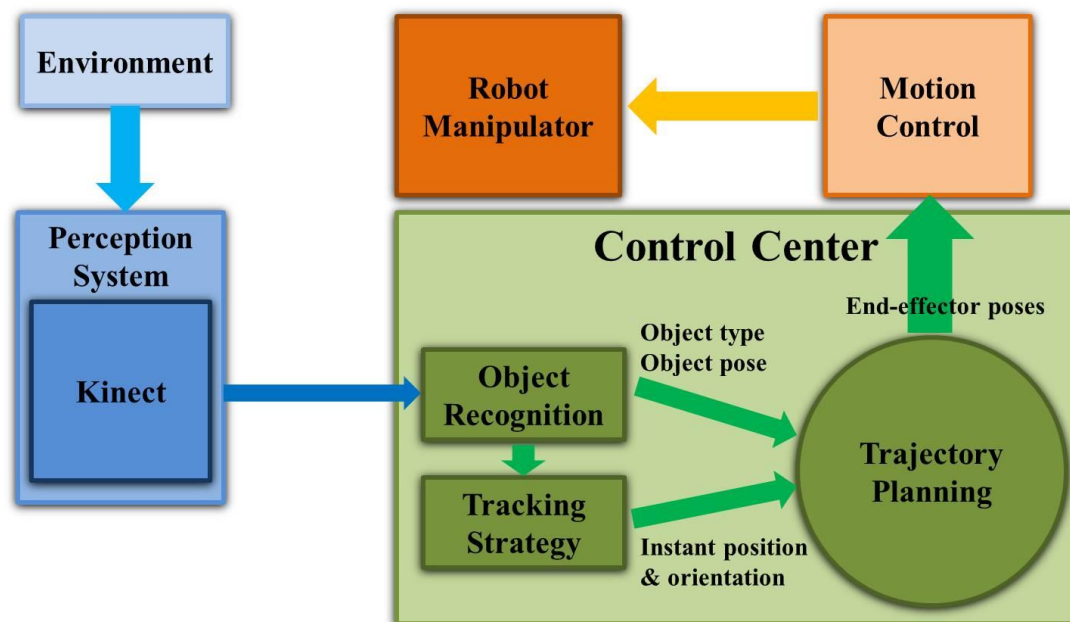
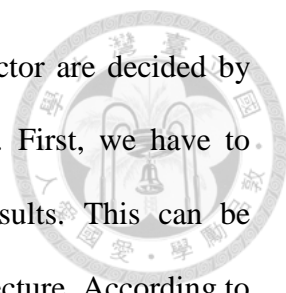


Fig. 3.2 Flow chart of specialized robot fetching system architecture.

Why it is called specialized? This means the model functions in specific situation.

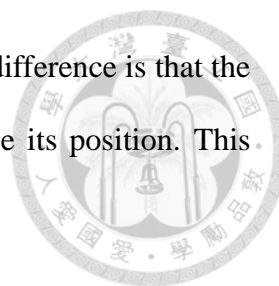
In this thesis, the main contribution will focus on the tracking strategy. Therefore, this

doi:10.6342/NTU201703382



specialized method is for tracking objects. Poses of robot end-effector are decided by 3D sensor data. The poses will be separated into two conditions. First, we have to collect static poses for each element from visual recognized results. This can be determined by Kinect sensor in the generalized robot fetching architecture. According to the assumption explained in subsection 2.3.1, the workspace will not change when the robot is performing its tasks. As a result, all we have to do is define some trajectory via-points, along which the robot will not hit itself or collide with the fixed environment. Since the environment is fixed, this trajectory always applies and can be stored in a database for later use. Since the environment is fixed, we can guarantee that this trajectory will never collide with the environment afterward. Therefore, the object type and pose will be sent to trajectory planning to calculate the static grasping pose and command the robot.

The task would become tough after objects are moving on the conveyor. In the same way, the environment is sensed by 3D sensors that describe the condition happening on the conveyor. There are two way to receive information from sensors: eye to hand and eye in hand. In this research, the sensor used in eye to hand is Microsoft Kinect sensor and the other one is webcam camera on the robot tooltip. Actually the moving pose derived from visual is a certain point in the camera coordinate. This will be transferred to the point under Cartesian Space coordinate. After tracking block finishing the computation, the instant state of the object, including its pose and orientation, will be transmitted to the trajectory planning. The robot manipulator then reaches to designated position to get ready to catch the object. In general, information from perception system will update the state of objects to the control center after the end-effector poses have been decided. Therefore, robot arm is able to grasp moving objects successfully.



The method in eye in hand is under the same architecture. The difference is that the end-effector will follow the position in the webcam view to change its position. This behavior seems to be what tracking it is.

The detail of our system is described in the Chapter 6.

3.3 Modified Robot Fetching Architecture

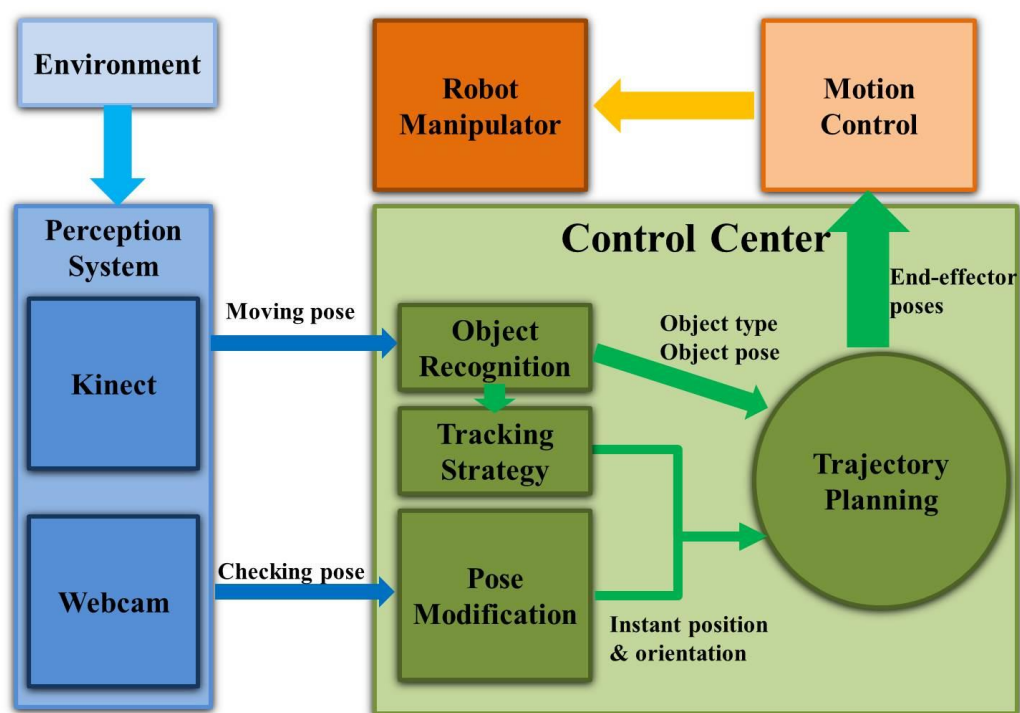
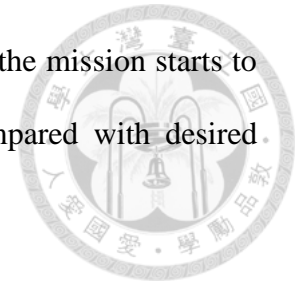


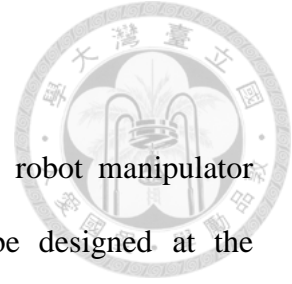
Fig. 3.3 System structure for modified robot fetching.

Actually, the main goal in this thesis is to grasp an object before it falls down to the ground. It seems that holding the target is success action in the result. However, to check the condition precisely, the grasping pose has a little difference from the desired command. If there is an offset or deviation angle, the task hasn't been completed yet. To check out whether the task is successful or not, the webcam is used to record image while the gripper is holding the object. The conceptual structure is shown in Fig. 3.3. To find out any variation of grasping pose, webcam will first store lots of correct pose

picture in a database. After gathering the poses as a standard level, the mission starts to grasp moving target. In this step, the grasping pose will be compared with desired command. The error model will be told in subsection 6.3.3 in details



Chapter 4 Manipulator



In this thesis, all algorithms are well-functioned only if the robot manipulator performs good behavior. How robot move and grasp should be designed at the beginning. Therefore, the main body will be fully introduced in its hardware structure and software functionalities.

4.1 Mechanism



Fig. 4.1 iCeIRA arm one in iCeIRA lab

The 7-DoFs of robot manipulator is shown in Fig. 4.1 in our lab [14]. The hardware has been built up by an alumnus of our lab three years ago. In this part, we will put emphasize on the detail pieces of robot, kinematics, control architecture and the software functionalities. The readers are encouraged to read the in-depth discussion on the hardware in the reference thesis [14]. The specifications of the robot manipulator are listed in Table 4-1. The payload is 5 kg without end-effector. However, the weight of 3 finger gripper is 2.3 kg. Thus, the actual payload is at most 2.7 kg, which need to be noticed. The weight of experimental elements is at the range of 225 to 350 g.

doi:10.6342/NTU201703382

Payload	~5 kg continuous and 15 kg instant (end-effector excluded)
Maximum speed	~50cm/sec (tool tip in Cartesian space)
DoF	7 revolute joints
Weight	~18kg
Working envelope	~900mm
End-effector	3 finger gripper

Table 4-1 spec of the iCeIRA arm one

4.1.1 D-H parameters

To begin with, the model of robot manipulator to describe its hardware in a standard way is illustrated by the D-H parameter method. Jacques Denavit and Richard Hartenberg introduced this convention in 1955 in order to standardize the coordinate frames for spatial linkages [15]. To be precise, 4 parameters are needed to describe the relationship between adjacent links and joints. They are θ , d , a and α . In Fig. 4.2, the procedure for attaching coordinate on the link and identifying these 4 parameters are described as follows:

Link frame attachment procedure:

1. Identify the joint axes and imagine (or draw) infinite lines along them. For step 2 through 5 below, consider two of these neighboring lines (at axes i and $i + 1$).
2. Identify the common perpendicular between them, or point of intersection. At the point of intersection, or at the point where the common perpendicular meets the i^{th} axis, assign the link-frame origin.
3. Assign the \hat{z}_{i-1} axis pointing along the i^{th} joint axis.

4. Assign the \hat{x}_{i-1} axis pointing along the common perpendicular, or, if the axes intersect, assign \hat{x}_{i-1} to be normal to the plane containing the two axes.
5. Assign the \hat{y}_{i-1} axis to complete a right-hand coordinate system.
6. Assign frame $\{0\}$ anywhere in the supporting base as long as the \hat{z}_0 axis lies along the axis of motion of the first joint. The last coordinate (frame $\{0\}$) can be placed anywhere in the end-effector as long as the \hat{x}_n axis is normal to the \hat{z}_{n-1} axis.
7. Try to assign the link frame so as to cause as many linkage parameters as possible to become zero.

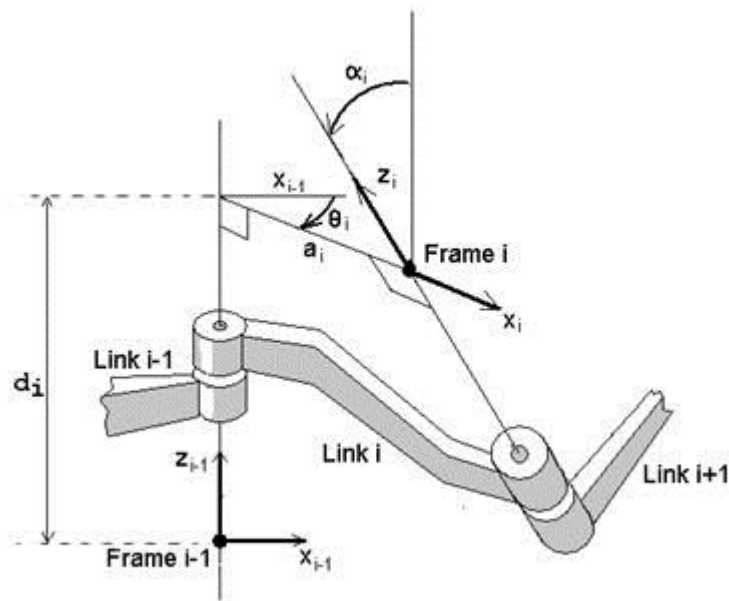


Fig. 4.2 The definition of four parameters

D-H parameters in terms of the link frames:

- d_i : the distance from \hat{x}_{i-1} to \hat{x}_i measured along \hat{z}_{i-1}
- θ_i : the angle from \hat{x}_{i-1} to \hat{x}_i measured along \hat{z}_{i-1}
- a_i : the distance from \hat{z}_{i-1} to \hat{z}_i measured along \hat{x}_i



- α_i : the angle from \hat{z}_{i-1} to \hat{z}_i measured along \hat{x}_i

After following the rule of D-H parameters, it is common to assign the link frames to identify the model of iCeIRA arm one. Each frame, coordinate definition and the D-H parameters are shown in Fig. 4.3. When all frames are defined in a proper solution, there is a D-H table to figure out the relationship between each revolute joint and the joint limit for the movement. The statistic is listed in Table 4-2.

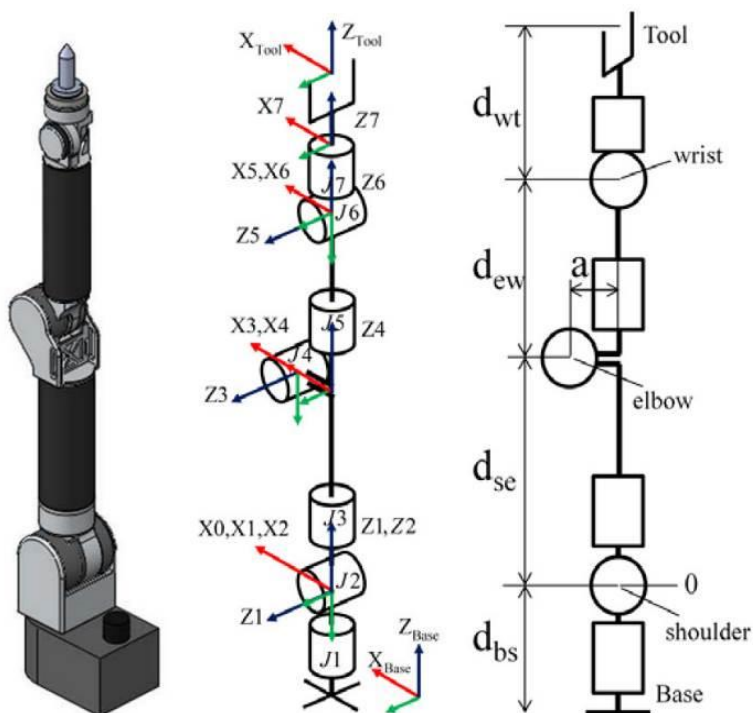


Fig. 4.3 Link frames assignment of iCeIRA arm one

After we receive the table, the DH transformation matrix from one coordinate from to the next is well implemented. Using a series of D-H Matrix multiplications, the final result is a transformation matrix from desired frame to initial one.

$$\begin{aligned}
 M_{DH} &= R_{z,\theta_i} T_{z,d_i} T_{x,a_i} R_{x,\alpha_i} \\
 &= \begin{bmatrix} c_{\theta_i} & -c_{\alpha_i} s_{\theta_i} & s_{\alpha_i} s_{\theta_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\alpha_i} c_{\theta_i} & -s_{\alpha_i} c_{\theta_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4-1)
 \end{aligned}$$

Joint i	$\theta_i(\text{rad})$	d_i	a_i	α_i	θ_u	θ_l
1	$\theta_1 + 0$	d_{bs}		-90°	180°	-180°
2	$\theta_2 + 0$			90°	120°	-90°
3	$\theta_3 + 0$	d_{se}	α	-90°	180°	-180°
4	$\theta_4 + 0$		$-\alpha$	90°	180°	-45°
5	$\theta_5 + 0$	d_{sw}		-90°	180°	-180°
6	$\theta_6 + 0$			90°	100°	-100°
7	$\theta_7 + 0$	d_{wt}		0°	360°	-360°

θ_u : joint angle upper bound

θ_l : joint angle lower bound

Table 4-2 D-H table of iCeiRA arm one

4.1.2 Transmission and actuator

The hardware structure of the one of the joints is shown in Fig. 4.4. To build up a whole manipulator, the key component including motors, drivers and gear box is indispensable. Those devices are shown in Fig. 4.5. The Maxon DC motor is chosen due to its excellence in the smaller size and high torque output. The dedicated motor driver is capable of current control, which indicates that we can directly control the torque of each motor. The gear box has two parts: one is gear set and another one is harmonic driver. The output torque from the motor is transmitted through a gear set, harmonic drive and then the joint. The advantages of harmonic drive are that they have compact size, high reduction ratio, and no backlash so that the size and weight can be reduced significantly. The detailed specifications on the reduction ratio and the motor are listed in Table 4-3. By means of those devices, the iCeiRA arm one now start to build up its control system.

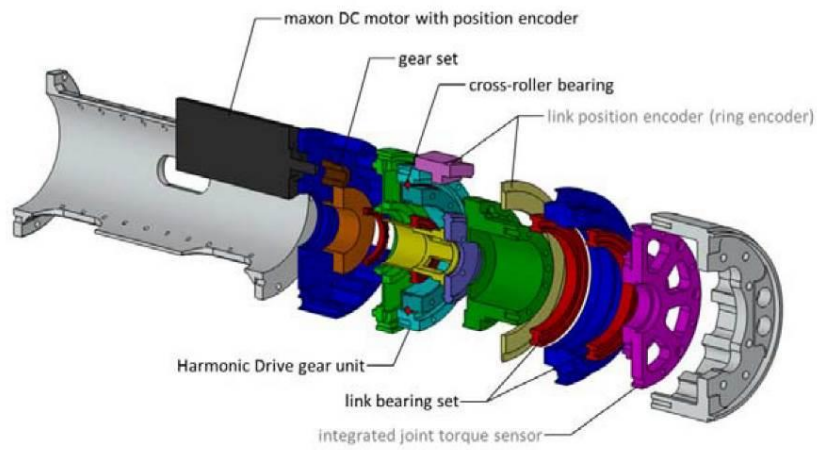


Fig. 4.4 Joint structure of robot arm



Fig. 4.5 The figure of motor, driver and gear box

Joint i	Gear set	Harmonic drive unit	Total gear rotation	Maxon motor	Motor nominal torque (mNm)
1	3:1	SHD-25-160 (161:1)	483:1	RE50 200w	354
2	3:1	SHD-25-160 (161:1)	483:1	RE50 200w	354
3	4:1	SHD-20-100 (101:1)	404:1	RE50 200w	170
4	4:1	SHD-20-100 (101:1)	404:1	RE50 200w	170
5	4:1	SHD-17-100 (101:1)	404:1	RE50 200w	85
6	4:1	SHD-17-100 (101:1)	404:1	RE50 200w	85
7	4:1	SHD-17-100 (51:1)	504:1	RE50 200w	28.8

Table 4-3 Transmissions and actuators

4.1.3 Gripper

The iCeIRA arm one is equipped with a Robotiq 3-finger gripper [16], which is shown in Fig. 4.6. This is the end-effector for object fetching system in industrial assembling application. This type of gripper owns two type of grasping: encompassing grip and fingertip grip. In our research, the fingertip grip is selected for grasping.



Fig. 4.6 Robotiq 3-finger gripper

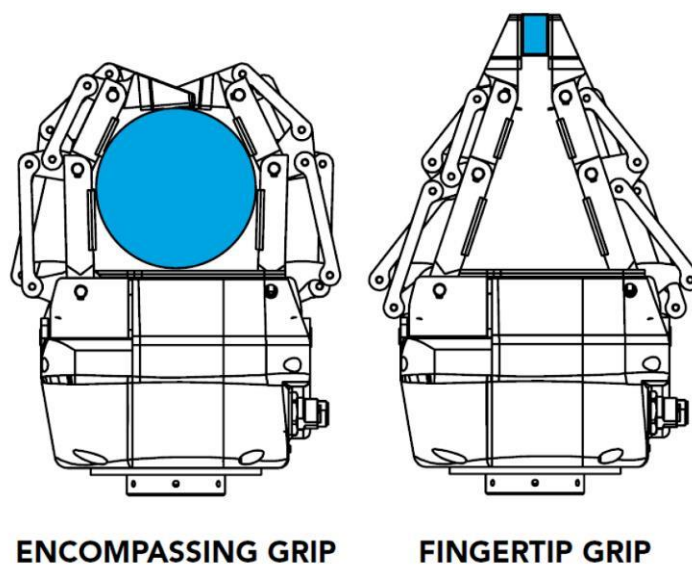
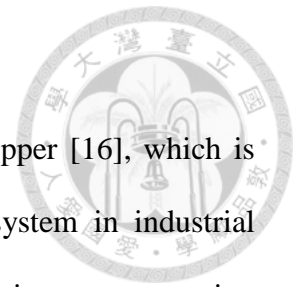


Fig. 4.7 Different type of grip



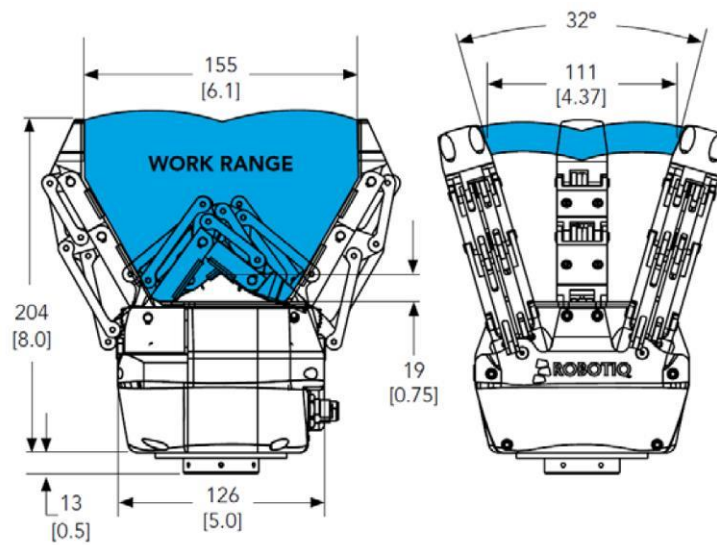


Fig. 4.8 Dimension and workspace of gripper

DoF	4
Weight	2.3 kg
Payload for encompassing grip (Fig. 4.7 left) : 10 kg	10 kg
Payload for fingertip grip (Fig. 4.7 right) : 2.5 kg	2.5 kg
Force	15~60N
Object diameter	20~155mm

Table 4-4 spec of gripper

The specification of the Robotiq 3-finger gripper is listed in Table 4-4. The dimension and the workspace of the gripper are shown in Fig. 4.8. The gripper is able to open no more than 155mm width. There are rubbers stick on the inner contact surface of each finger to increase the friction between the finger and the grasped objects. The friction coefficient of the rubber is 0.4.

We choose this 3 finger gripper because of its adaptiveness. The special linkage design of the mechanism makes it compliant to the shape of its holding object.

Furthermore, the motor will halt once the resistant force is beyond a given value, which

doi:10.6342/NTU201703382

is good for grasping objects of unknown shape without damaging it. Once the motor is halted due to the external resistant force, it will not continuously exert force, which may result in over-heating of the motor. On the other hand, the current position will still be maintained by the linkage mechanism.

There are 4 operating modes: basic, pinch, wide and scissor mode as depicted in Fig. 4.9. The basic mode is commonly used mode and is suitable for most of the cases. It is especially, suitable for cylindrical or stick sharp objects. The wide mode is designed for holding large or round object. The pinch mode and scissor mode are used for picking up small objects precisely. The scissor mode is even more precise than the pinch mode at the expense of payload and adaptiveness. These 4 operation modes are sufficient to cover most of the use cases. If the user requires a more dexterous manipulation, it can also control each finger individually. In this thesis, the basic mode and pinch mode are the main operation to finish grasping action.



Fig. 4.9 Four operation modes of the gripper

4.2 Control Architecture

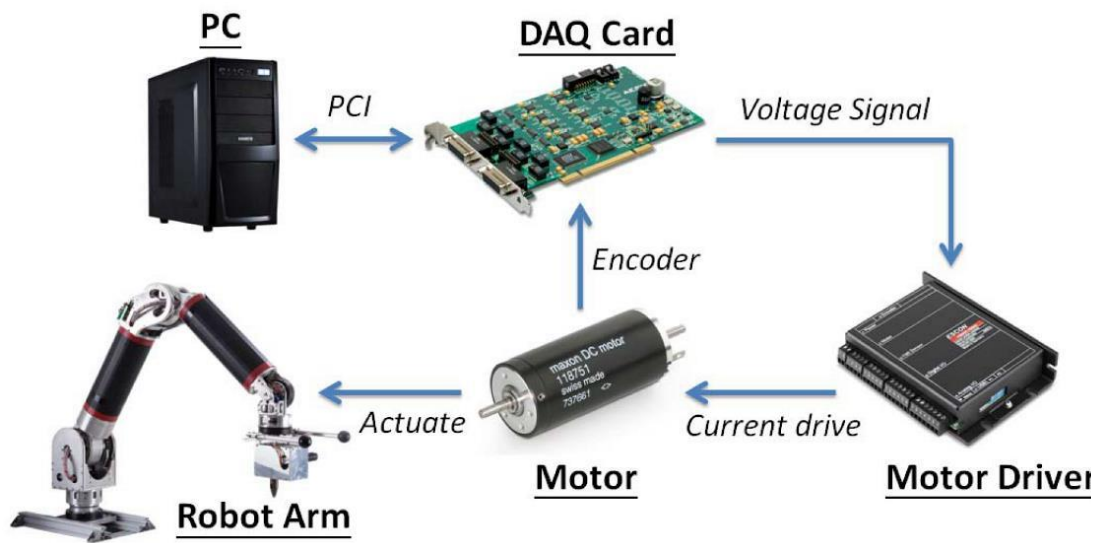


Fig. 4.10 Control architecture in robot control system

The control architecture of iCeIRA arm is shown in Fig. 4.10. A PC running in RTOS is the control center of the whole system. It communicates with the hardware components such as the motor driver and the motor encoder through a DAQ card installed on the PCI bus. The RTOS is required since we would like to do the motor position control on the PC running at 1 kHz.

The DAQ card is installed in the PCI bus of the PC, which endows the PC with the capability of sending an analog voltage signal to control the motor driver and reading the encoder count from the motor. On our platform, one PISO-DA8U [17] for 8 channel analog output, and two PISO-Encoder600 [18], each for 6 channel encoder count, are chosen.

When received the analog voltage signal from the DAQ card, the motor driver will output a corresponding current to drive the motor. This implies that the current control loop is done internally by the motor driver. For example, when the driver receives 10V signal, it will output the positive maximum current, which can be set using a variable

resistor on the driver. The unit of the drive current is in Ampere. On the other hand, when the motor receives -10V signal, it will output the negative maximum current; and when it receives 0V, it will output zero current. In our case, Maxon ESCON 70/10 is chosen, which support maximum output power of 700W and maximum output current of 10A.

The motor is driven by the current from the motor drive and output a torque to actuate the robot arm. The output torque equals to the product of the input current and the torque constant of the motor. The rotating angle will be count by the encoder equipped on the motor and then feedback to the PC through the DAD card. This closes the loop so that the accurate position or velocity control can be done on the PC.

4.3 Software Architecture

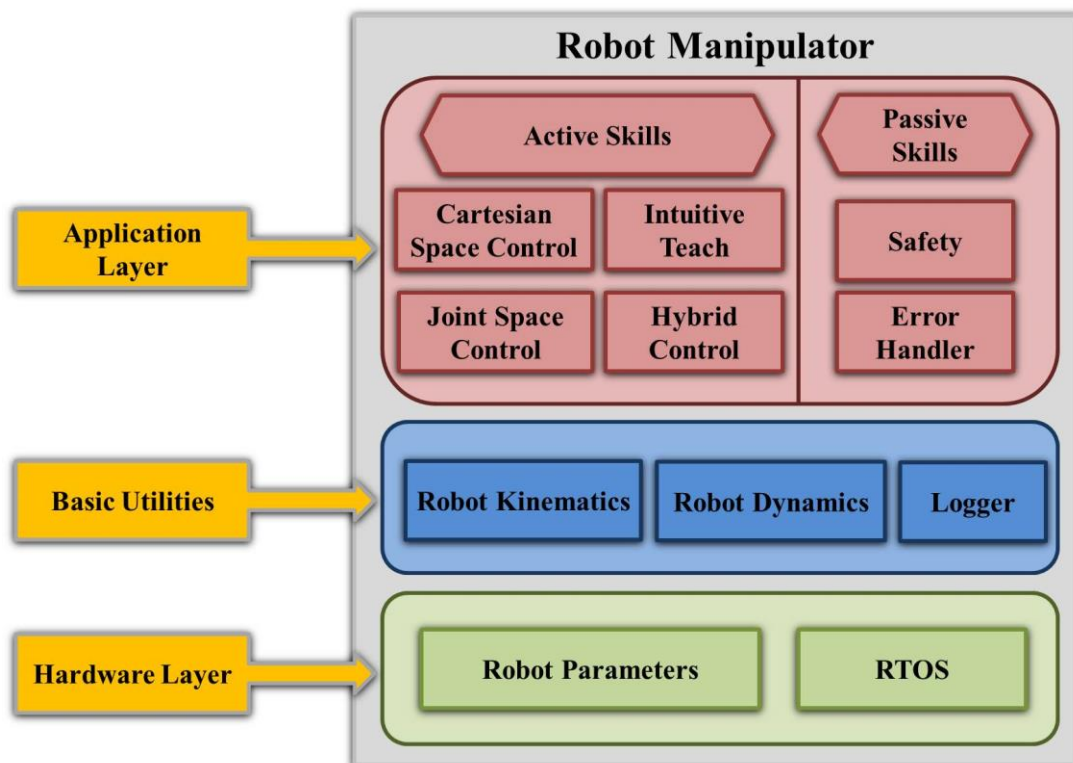


Fig. 4.11 Software architecture

The software architecture of the robot manipulator is illustrated in Fig. 4.11. This

doi:10.6342/NTU201703382

architecture is divided into three layers: application layer, basic utilities and hardware layer. In this section, the motivation why to build this architecture will be described. The functionalities in each layer will also be explained.



4.3.1 Motivation

The motivation for proposing this software architecture and rebuild the whole software for the robot is that we require a clearly defined and modularized architecture for future development and maintenance. The first version of the robot's software was built by an alumnus of our lab 3 years ago, who is also the designer of the iCeIRA arm one mechanism. We acknowledge and appreciate his effort in building the mechanism and also the basic software utilities of the robot arm such as motor control, trajectory planning, intuitively teaching by touching, and so on. However, the software part hasn't been developed yet. To made the maintenance of programming, there layers is created to illustrated the functionalities of robot manipulators and extend the capabilities of iCeIRA arm one.

4.3.2 Hardware Structure

The hardware layer is the abstraction of the hardware components. The abstraction of the robot such as the D-H modes, controller parameters, dynamic parameters, etc, is encapsulated in the Robot Parameter module. The abstraction of the PC are divided into two modules, one is the RTOS, which provides a precise real time timer, and the other is the communication interface, which can be DAQ card, RS 232, CAN-BUS, EtherCAT [23 24], etc. this layer strongly depends on the hardware. For different robots, the functionalities of this layer can support might change drastically. As a result, we would not try to standardize this layer and unify the interface of each module. Instead, the programmer should try to identify the hardware dependent parts and classify them in

this layer so that other modules would depend less on the platform which makes porting this program onto other platform easier.



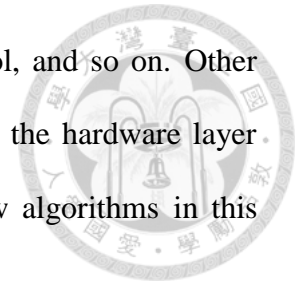
4.3.3 Basic Utility

This layer provides the basic utilities for building up robotic applications. For example, the robot kinematics and dynamics are the most fundamental utilities that almost all the robotic applications will require. The logger module is for recording various states of the robot arm such as joint position, joint velocity, joint torque, tool tip Cartesian space position, Jacobian, etc. The logger module will dump the recorded states into the hard disk for latter analysis. Some other modules such as the Joints, Sensors, and Grippers also belong to this layer. The programmer will not have to handle the low-level communication interface signal to control the motor or retrieve the sensor data. On the contrary, they can use the abstract Joints and Sensors modules to complete their task more conveniently. This layer is partially dependent on the platform mostly because of the Joints, Sensors, and Gripper module. These modules usually have strong dependency on the communication interface. For example, the joints might be connected in series and communicate with EtherCAT and sometimes the sensor signal is retrieved from a DAQ card installed on the PC's PCI bus.

4.3.4 Application Layer

The purpose of the two aforementioned layers is to provide most of the necessary utilities and a proper abstraction for high-level robotic applications. With these two layers, developers can build their own application in the top layer – the application layer. This layer is independent of the hardware. It is more like a collection of the fancy functionalities. Most of the programmers can develop their own algorithms in this layer and don't have to get their hands dirty. Handy, basic and powerful utilities are provided

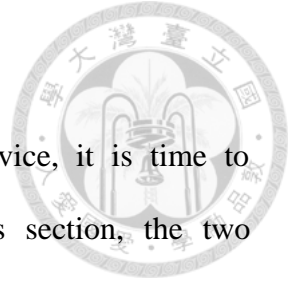
in the second layer such as the kinematics, dynamics, joint control, and so on. Other programmers can harvest the hard work which have been done in the hardware layer and the basic utility layer, and do their research and develop new algorithms in this layer.



We further divide this layer into two different groups – the active skill and the passive skill. The former one refers to services that operate only when they are called, while the latter one will operate periodically behind the scenes. One of the examples for the active skill is the Cartesian Space Control, which is the control of the end-effector from its current pose to a give target pose. The robot will have to run the service only when it gets the order to go to some target pose. On the other hand, the Safety, which should detect the collision between the robot and the environment, belongs to the passive skill since the robot will have to check the whether the collision occur periodically.

4.3.5 Timer

In this architecture, three threads are created to perform different tasks. The active skills and the devices such as the Joints, Sensors, and Grippers, are updated in the first thread. This thread is a real time thread which will update at 1 kHz. Among all the thread, the first thread is most time-critical. It involves the update of the motor feedback command, sensor data, and current performed task. The second thread updates the passive skills and the robot states. This is also a real time thread in order to retrieve the latest robot state and the immediate reaction to safety issues and internal errors. The third thread is for the logger. This thread need not be a real time thread since it performs lots of data read/write on the hard disk. The task is neither time nor safety critical. Therefore, a relatively resource consuming real time thread is not required for this operation.



4.4 Manipulator Functionalities

After describing the software architecture and hardware device, it is time to develop the basic functionalities of robot manipulators. In this section, the two techniques are essential for iCeIRA arm one to implement assembling task for industrial application.

4.4.1 Intuitive teaching by touching

It is the user's responsibility to tell the manipulator where to go. Usually, we concern only about the pose of the end-effector in Cartesian space since it is the end-effector that interacts with the environment. In industrial application, the operator will record a series of end-effector poses and the manipulator simply replays these poses when performing its tasks. Therefore, it is the operator's responsibility to record the series of poses precisely so that the manipulator can finish its task successfully. Traditionally, the operator will command the robot to go to a target pose using the teaching pendant or direct key in a pose. The operator will fine tune the pose until the robot reach the desired target pose and then record the current pose. This process is actually tedious, time-consuming and counter-intuitive. Therefore, here we introduce the teaching by touching [19][20] so that the operator can guide the robot direct and teach the robot intuitively and quickly. The teaching by touching enables the operator to move and guide the manipulator directly by hand. Therefore, the operator can guide the robot to the desired pose and fine tune the pose intuitively.

Generally, it is impossible and not allowed to guide the robot. That is to say, it is difficult to move the robot to another pose by hand. The major reason is that the motor will try to maintain the current pose so that it will exert torque to resist external forces. Some minor reasons include the high reduction gear, friction in the transmission,

mechanism weight, etc. to solve the major problem, the manipulator will have to sense the direction of the external force and comply with the external force instead of resisting it. In our case, we did not rely on an additional sensor such as the force torque sensor. Alternatively, we compute the direction of the external force by the following formula:

$$\delta x = J(q)\delta q \quad (4-2)$$

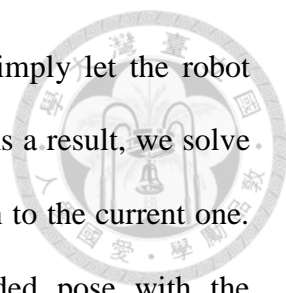
, where δx is the slight variation of the tooltip's pose in Cartesian space, which can be defined as

$$\delta x = \begin{bmatrix} \delta x \\ \delta y \\ \delta z \\ \delta \alpha \\ \delta \beta \\ \delta \gamma \end{bmatrix} \quad (4-3)$$

, where $\delta \alpha, \delta \beta, \delta \gamma$ are the rotation angle relative to reference coordinate of x, y, z axis. The δq is the slight variation of the joint angle measured from the encoder error, and the $J(q)$ is the Jacobian matrix calculated from current joint angle q . As a result, δx can be regarded as the direction of the external force. As for the minor problems, gravity compensation and the friction compensation must be included to resist the mechanical weight and the friction respectively. The gravity compensation is calculated from the mass center to maintain static equilibrium. The friction compensation is a constant to resist the static friction of each joint.

Besides the basic intuitive teaching by touching functionality, we further decouple the guided motion into translation and rotation. In the translation mode, the orientation of the tooltip is fixed while the position can be guided freely by the operator. On the other hand, in the rotation mode, the position of the tooltip is fixed while the orientation can be guided freely.

In the translation mode, once the encoder error is above a threshold, the robot will



move along the direction of the external force. However, if we simply let the robot moves along that direction, the orientation will change inevitably. As a result, we solve the inverse kinematics for the new pose while setting the orientation to the current one. This equivalently makes the manipulator moves along the guided pose with the orientation fixed. The same idea applies to the orientation mode in which the position of the tooltip is fixed while the orientation can be guided by the operator freely.

4.4.2 Online trajectory generation

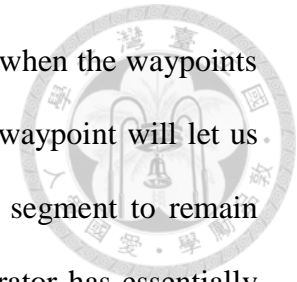
The generation of command variables for industrial manipulator has two functions: specification of the geometric path (path planning) and specification of the progression of position, velocity, acceleration, and jerk in dependence of time (trajectory planning). The literature provides many of approaches and algorithms in both fields.

The goal of trajectory planning is generating smooth motion in multidimensional space refer to a set of planned waypoints or path points by using some given parameter, i.e. the time interval of a motion, boundary conditions of motions, the maximum velocity limit, acceleration limit or even jerk limit. The smooth motion is a function of time that is continuous and has a continuous first derivative. Sometimes a continuous second derivative is also desirable. Jerky motion tends to cause vibration in the manipulator and cause increased wear on the mechanism. There are many methods about trajectory generation have been used in many systems, such as the cubic spline line, B-spline [21], tension spline [22], the linear function with parabolic blends and NURBS [23]. The above methods generate trajectory that consist by numbers of segments which are functions of time and smooth connect each segments by matching the boundary conditions. Then the trajectory can be generated at run time, that is, the trajectory points can be computed at each servo loop or sampling time in control system. These methods are useful in the case when the waypoints of trajectory are pre-known or

predictable so that we can compute it offline. However, in the case when the waypoints are unknown and unpredictable, the frequently change of the next waypoint will let us need to re-calculate a new computing equation of next trajectory segment to remain smooth motion in real-time. There is other type of trajectory generator has essentially online calculation. An online trajectory generator is a real-time trajectory planning algorithm for computing the interpolation of synchronized and time-optimal smooth motion in multi-DoFs with arbitrary input values. The next trajectory set point is computed according to current state of motion every control cycle, typically every millisecond. This enables system to react instantaneously to unforeseen and unpredictable event, usually sensor triggered event, at any time instant and in any state of motion.

The interface of an OTG has basic form shown in Fig. 4.12 and the OTG have the following specifications:

- The input values for the trajectory generator are completely arbitrary. Expect the motion state \vec{p}_{i-1} , \vec{v}_{i-1} , \vec{a}_{i-1} of the last control cycle, all values may change between the control cycles. This means that This means that the target position \vec{p}_i^{trgt} , (the target velocity \vec{v}_i^{trgt}), the max. velocity \vec{v}_i^{max} , the max. acceleration \vec{a}_i^{max} , and the max. jerk \vec{j}_i^{max} are not constant nor continuous.
- The trajectory is calculated on-line (in real time during every control cycle), because the input values may change unpredictably. Only the next sample point, the \vec{p}_i , \vec{v}_i , and \vec{a}_i is calculated within one control cycle i .
- Synchronization: the OTG consider an N-dimensional space, where N is the number of DoFs. The input and output are $N \times 1$ vectors in the space. For the manipulator, it is in joint space or Cartesian space. Synchronization is important requirement, that is, all N DoFs have to reach their target position simultaneously.



at zero velocity and zero acceleration. Furthermore, for the straight line motion of manipulator's end-effector, the phase-synchronization is required so that all N DoFs will not only reach their target at same time but also change the state of motion simultaneously. For example, all N DoFs change their motion state from acceleration to constant velocity at same time.

- The generated trajectory for the DoF with the largest execution time is time-optimal. The constraints for other DoFs are adapted for synchronization.
- An OTG do not consider path planning. The desired target position \vec{p}_i^{trgt} is user-given.

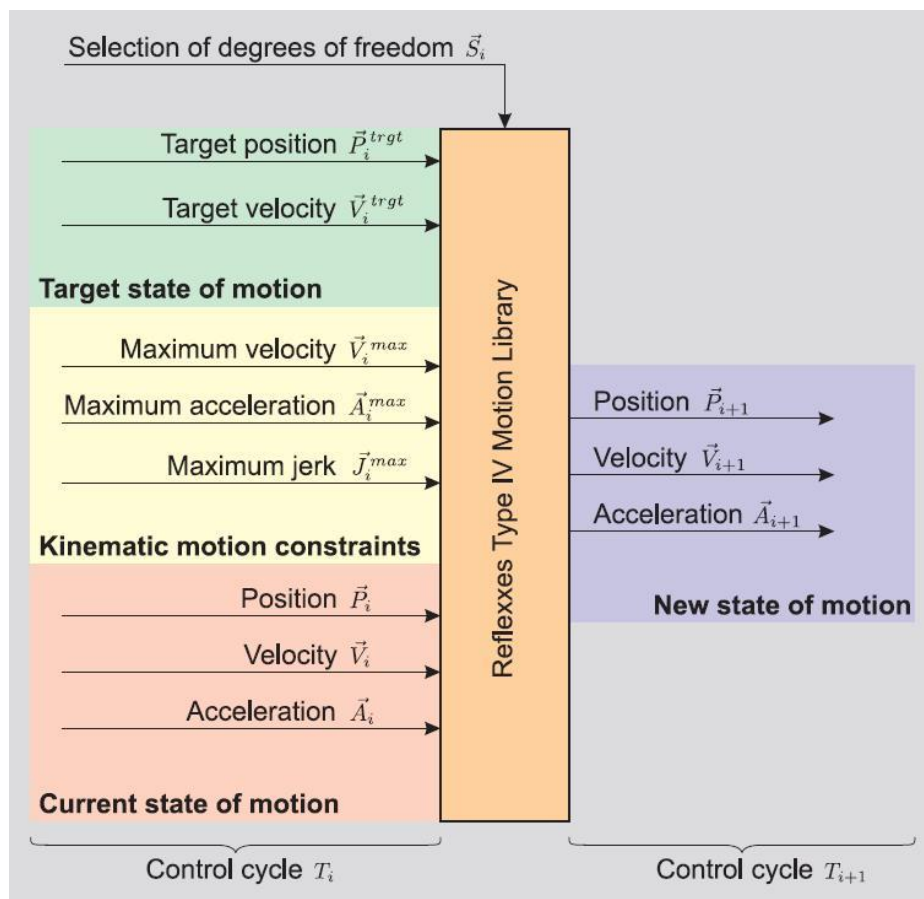
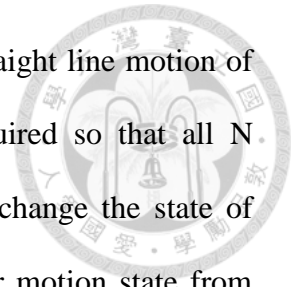
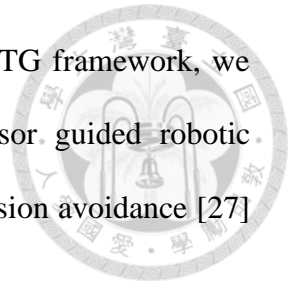


Fig. 4.12 The input and output of an OTG algorithm

In recent times works on OTG have been published, e.g. [24][25]. A new control scheme that uses an OTG algorithm as an interface between sensor process and



low-level robot motion control has been proposed. Based on the OTG framework, we can simply deal with the unforeseen sensor events for the sensor guided robotic applications such as visual servo control [26] and human-robot collision avoidance [27] of robot arms. It consists of three layers:



- 1) A sensor processing algorithm used to compute a desired pose or other motion states for the robot.
- 2) The desired motion states are used by the OTG algorithm to instantaneously compute a motion trajectory that connects to the current state of motion.
- 3) The output signals of the OTG algorithm are the motion command of a trajectory-following motion controller.

Due to the intermediate layer, a number of advantages are achieved:

- Jerk-limited and continuous motions are guaranteed independently of image processing signals. Acceleration and velocity constraints due to limited dynamic robot capabilities can be directly considered. Physical and/or artificial workspace limits can be explicitly applied.
- In cases of sensor failures or inappropriate image processing results, deterministic and safe reactions and continuous robot motions are guaranteed.
- The image processing does not necessarily have to be real-time capable.
- High performance due to low latencies, because motion trajectories are computed within one low-level control cycle (typically one millisecond or less).
- The architecture is of a very simple nature and can be integrated in many existing robot motion control systems.

Chapter 5 Object Recognition

In this chapter, the object recognition system will be introduced



5.1 Point Cloud Library

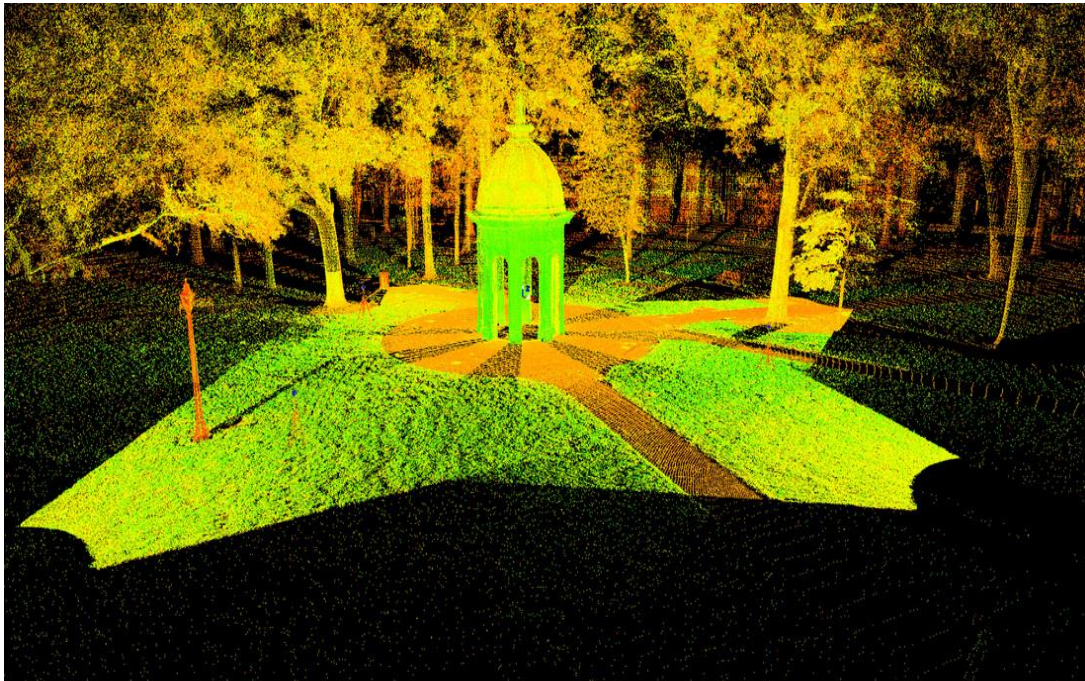
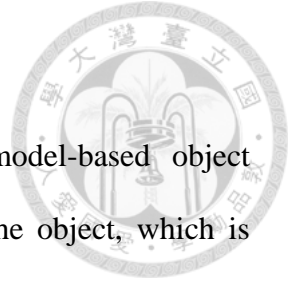


Fig. 5.1 Point Cloud illustration

The Point Cloud Library (PCL) is a standalone, large scale, open project for 2D/3D image and point cloud processing [28]. For a 2D image, the frame shown on the screen is a kind of sampled image by pixels. All information in the environment is projected on the image 2D plane. Similarly, a Point Cloud, as shown in Fig. 5.1, is the sample of the environment with points. Each point contains (x, y, z) 3D information in Cartesian space and sometimes even the (r, g, b) color information. The PCL turns out to be a large collaborative effort in robotics, especially being used as visual servo feedback control in the surroundings. Besides, the PCL is also programmed by Robot Operating System (ROS) [29].



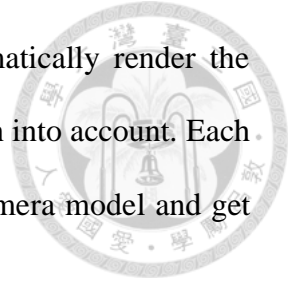
5.2 Database Generation

The global recognition pipeline can be classified as a model-based object recognition algorithm. As a result, we need the CAD model of the object, which is easily obtainable since the objects are manufactured according to the CAD model. Even though the models are easily to gain, some processing is required to construct the recognition database. The most important one is the partial views rendering. Imagine that the object is presented in the environment and grabbed by the 2.5D sensor such as Microsoft Kinect or ASUS Xtion, only the partial view can be retained. Because the back side cannot be seen by the camera, the camera is unable to generate the Point Cloud. As a result, if we want to match the grabbed partial view against the database, we must construct the database containing partial views around the object. That is to say, we should first generate partial views from the complete 360 degree model and store them in the data base [30][31]. The procedure is as follows:

1. Convert the CAD model to a mesh model
2. Load the mesh model into the OpenGL renderer [32][33]
3. Equally sample and transform the viewpoint around the model
4. Retrieve the depth map of the partial view from the depth buffer
5. Back-project the depth value of each pixel in the depth map into the 3D Cartesian space
6. Return the Point Cloud of the partial view

In the first step, the CAD model is converted into the mesh ply model. The first reason for doing so is that we only need the surface info of the model for rendering. The second reason is that OpenGL can only read in mesh models. In the third step, the position of the viewpoint is sampled using the spherical coordinate with radius equal to

one meter constant and for every 20 degree. OpenGL will automatically render the model and generate the depth map, which already takes the occlusion into account. Each pixel in the depth map can be back-projected using the assumed camera model and get its position (x, y, z) in the 3D space in step five.



5.3 Kinect Calibration

The vision sensor with PCL algorithm is Microsoft Kinect sensor. Besides, cameras play an important role in collision avoidance in order to find out the obstacle in the environment. There are lots of sensors to use for object recognition. Therefore, Kinect made by Microsoft [34] is the better choice in our research to implement a safety working space in the production line. 3D information can be derived from Kinect, especially depth data. With the help of 30 frames per second rate, varieties surrounding the robot can be easily detected. Furthermore, Kinect is equipped with two micro-motors which can alter the orientation of scene immediately. Human can set up this kind of sensor to obtain the precise visual angle for safety region. The bigger region a Kinect can provide, the safer and prompter avoidance arm will react to.

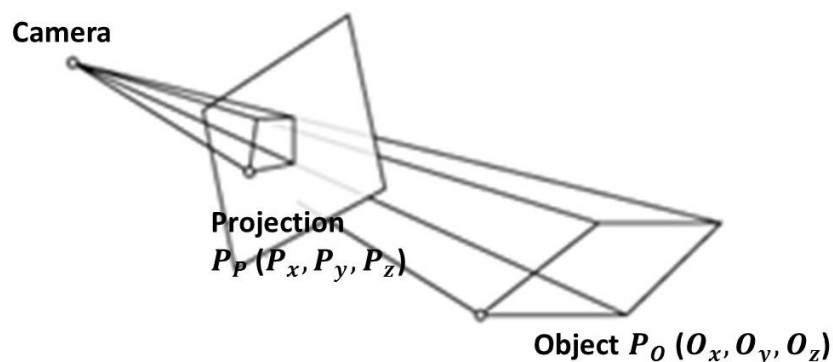


Fig. 5.2 The camera distortion on the projection plane

Although the advantages of Kinect, frame data should be revised because of non-homogeneous space in depth. In Fig. 5.2, object information in the work space will

project to image plane by Kinect. Therefore, object position O_x and O_y will be altered to P_x and P_y inaccurately while depth information will remain the same.

To modify position x and y to reference frame, the chess picture is used to calibrate the intrinsic matrix M_I and the extrinsic matrix M_E , which represent the transformation between the camera frame and robot coordinate.

$$M_I = \begin{bmatrix} S_x & 0 & C_x \\ 0 & S_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \quad (5-1)$$

where S_x and S_y the proportion of meter per pixel, C_x and C_y represent the center of image plane. Therefore, from Eq. (5-1)

$$P_x = \frac{O_x * S_x}{O_z} + C_x$$

$$P_y = \frac{O_y * S_y}{O_z} + C_y \quad (5-2)$$

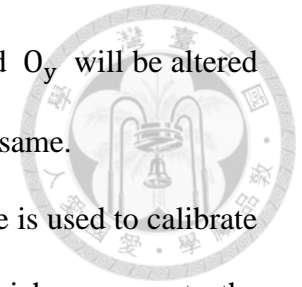
$$P_z = O_z$$

Later, the relation from Kinect to robot base is

$$P_R = M_E * P_P \quad (5-3)$$

where P_P is the object projection position in image plane and P_R is the position under reference coordinates. After completing the calibration, the accurate data information can be applied to robot manipulator.

The checkerboard camera calibration [35] and the Point Cloud Library (PCL) [36] are used in order to obtain the intrinsic matrix M_I and the extrinsic matrix M_E in real world for calibration. PCL, which originally developed by Radu B. Rusu et al, is an open source library for 3D Point Cloud processing. By using PCL, it is easy to translate the coordinate from Kinect to robot base. Then the extrinsic matrix M_E can be



calculated automatically by PCL shown as Fig. 5.3.

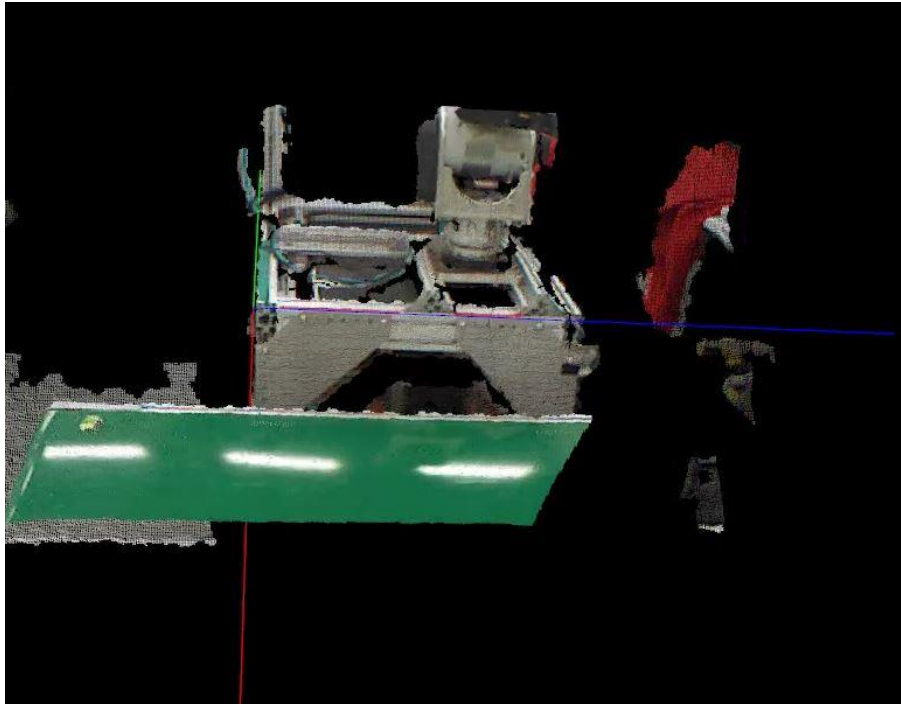


Fig. 5.3 The PCL translates the Kinect coordinate to reference coordinate

5.4 Object type and pose recognition

After using the PCL algorithm for object recognition system to build up the database, the target should be defined and described for pick and place task. Besides, the grasping pose for each target also be decided by robot control system. When the database is fully constructed, the command to grasp will be done by intuitive teaching by touching method.

There are three type of element in this research, and are shown in Fig. 5.4. Every type owns individual CAD model, which can be successfully recognized by PCL algorithm. These types should be stable poses because the Kinect only captures the instant pose to transfer data to the control center. It is possible to teach the robot with only a few and finite number of grasps so that the robot is capable of dealing with object with arbitrary stable poses. In this thesis, we use the intuitive teaching by touching

doi:10.6342/NTU201703382

functionality provided by the manipulator. However, we cannot simply store the taught grasp in the database. The taught grasp must be bound to the object coordinate so that it can be adapted according to the recognition result. If the pose changes dramatically in a period of time, the manipulator is unable to grasp it due to different recognized results. Thus, each stable pose owns only one grasping pose in the robot object fetching system.

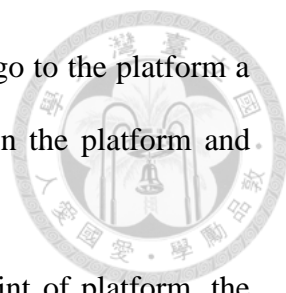


Fig. 5.4 Three types of objects in the pick and place task

The procedure of pick and place can separate into two parts: pick and place. The pick part contains three sub-steps: pre-pick, pick and post-pick. Similarly, the place part also includes pre-place, place and post-place. The previous part is like a middle point between initial pose to target pose, and so does the posterior one. All six steps are completed by intuitive teaching and touching. The whole pick and place procedures are as follows:

1. Pre-pick: This pose is generally a short distance away from the target object in case that the finger or the tooltip might collide on the object and change to pose of the object accidentally.
2. Pick: The gripper follow the command to the target directly above, and the finger will close to hold the recognized object
3. Post-pick: the arm will retreat backward to the point where a short distance away from the pick point.
4. Initial point: A transition state from pick step to place part.

doi:10.6342/NTU201703382

- 
5. Pre-place: This pose will follow different type of object to go to the platform a short distance above. This can prevent arm from run down the platform and keep safe before reaching to targets.
 6. Place: When the actual pose is the same as the middle point of platform, the end-effector will smoothly open and place the target.
 7. Post-place: After finish placing action, the manipulator will go to a transition point preventing from singular point or collision. And finally it will go back to the initial point to wait for next command.

Chapter 6 Object Tracking Strategy



To catch the moving objects from the assembly lines, it is useful to cooperate with sensory information to deal with this problem. In this thesis, the tracking architecture is a sensor-integrated system to find out where the object is instantly. A Microsoft Kinect sensor is placed beside the conveyor for object recognition system and tracking system. Another one is webcam camera, which is positioned at the end-effector. The position of two sensors relative to the robot manipulator is shown in Fig. 6.1. In later subsection, the tracking strategy will be introduced in details with the challenges and technique for modified movement.

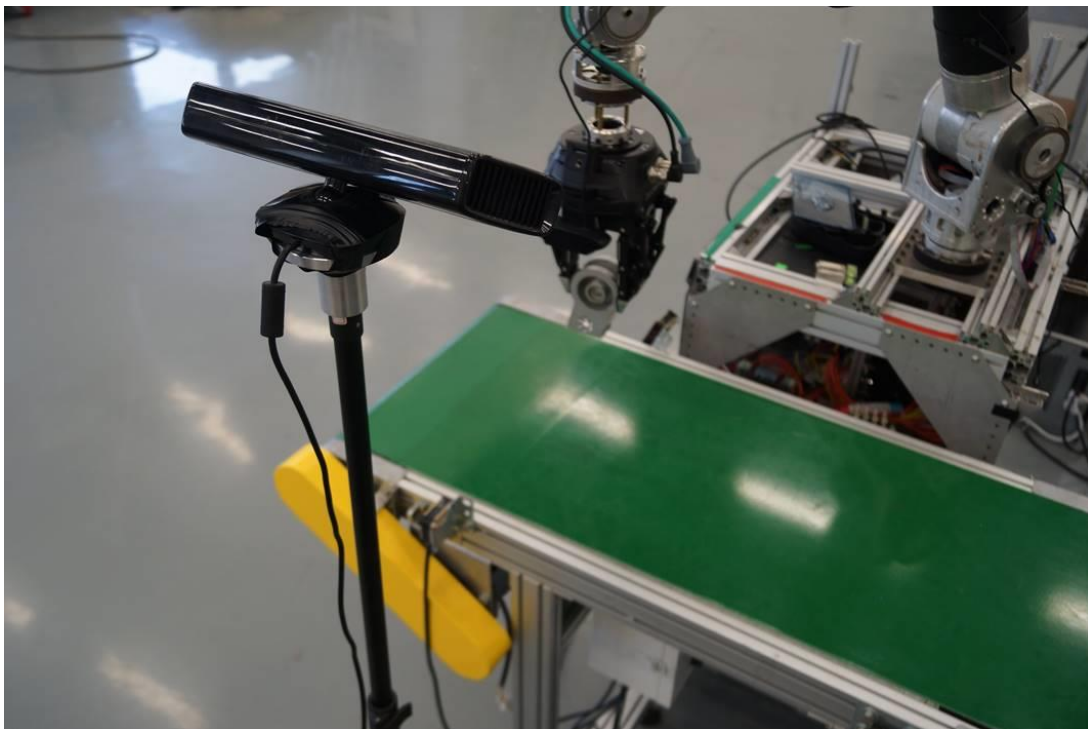


Fig. 6.1 The position of two sensors in experiment

6.1 Problem Statement

Actually, picking up a static object is common and easy because all data will remain the same after recognition finished. When the movement of object starts to

change, especially a straight motion on the conveyor, the main goal is to identify the object position in every sampling time and track the object over subsequent frames. The desired output is the updated and transferred information from visual sensors to the manipulator. However, there are frames in which the object is not present or is only partially visible. Therefore, the aim is to come up with a robust algorithm to detect and track the object on the conveyor under these constraints.

6.2 Tracking Algorithm

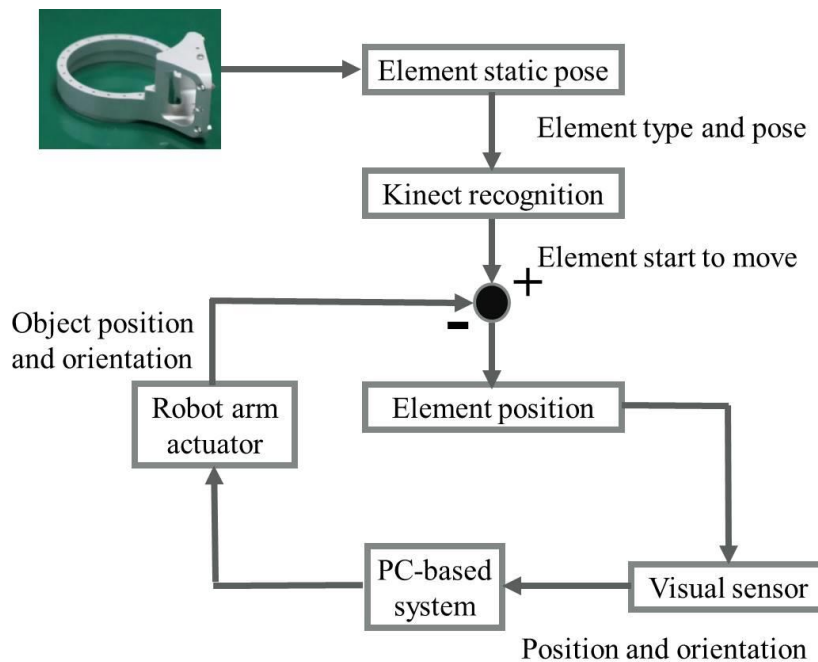
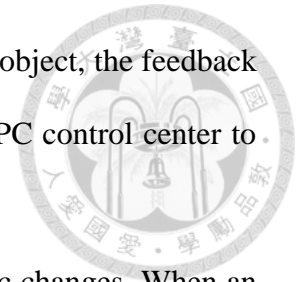


Fig. 6.2 The tracking diagram

In Fig. 6.2, the brief architecture of tracking is the feedback loop of object position and orientation. Picking operation has been implemented in the previous section. When the static pose of element is received from Kinect recognition, the conveyor will start to move. The visual sensor, which plays an important role in tracking, will collect the position and orientation of object from camera coordinate to robot Cartesian space coordinate. The data will update the command to grasp and robot arm actuator will

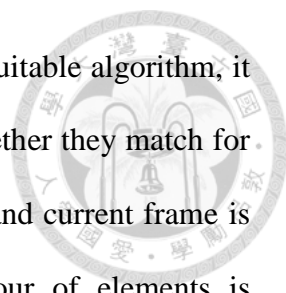
follow new one to reach the target. Before the end-effector hold the object, the feedback control system will continuously collect instant information to the PC control center to make sure the success.



Now the scenario tends to be more difficult with some dynamic changes. When an object starts to move on the conveyor, the information of it should be updated to the robot to implement tracing skill. There are some challenges of tracking technique under visual control system. First, we need to track the objects by using the image processing in real time frame by frame. We then take the previous ten frames as its average reference. Secondly, we need to define the position of the gripper contrast to the moving target. We thought of one strategy about how the position of the objects can be obtained. The Kinect that also perform recognition can provide us information about not only the target's color, but also their depth data. Then, we generate a method to get 3D coordinate with the collected information and calibrate the scale to y axis and z axis with the consideration of depth frame that being x axis under Kinect view. Lastly, robot reaction will follow the renew data to finish grabbing movement. There are three subsections in our tracking strategy. After we obtain previous ten frames as the reference to compare with the current frame, we generate a promising perspective to improve the position of objects by calibrating the scale of y axis and z axis. After we get the position of the tracker and the target, then new motion is derived, and the tracker in the end would follow the target according to the calculated trajectory.

6.2.1 Image segmentation

Because we want to display the tracking in real time, the algorithm of image segmentation must be simple and effective. After trying a lot of algorithms about tracking moving objects, we want to find one that can process the fastest in the time manner. Some algorithms take too long or lagging too much in updating frames, and

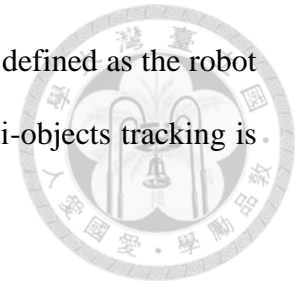


that may go against at initial goal in this research. If we think of a suitable algorithm, it is natural to continue on our next step in testing them, checking whether they match for us. In detecting objects: First, different intensity between previous and current frame is used to generate the contour of the object. However, the contour of elements is fragmented because the small size relative to the big conveyor is too small for detection, which may lead to some errors about finding the center of contour. In addition, the working area is limited by the vision of Kinect; we cannot speed up the conveyor to get the intensive difference data such that the threshold of finding the moving objects can't be high enough. Sometimes, the noise of Gaussian white noise would affect the contour. Hence, we could not use this way to find the objects. We choose the other way to generate the contour by recording the first ten frames as the average reference. The reason why we pick ten frames is that the Gaussian noise should be diminished as much as possible by get the average of the reference frames. This work can be shown by minimize the standard deviation with sampling. Then, we can get the difference between the reference and current frame, finding the contour. This segmentation result comes out to be very well. Both the end-effector and the shifting element are tracked.

6.2.2 Position localization

After the contours of objects are derived from the camera, we want to get the instant position of the tracker (gripper) and the target (element) separately. First, the contour is the difference between reference and current frame from the color frame. Afterward, we get the coordinate of the objects in depth frame. The desired data include situation along y axis and z axis from Kinect color frame and x axis in the depth frame. Because of the different x-y planes that the tracker and target belong to, we can separate the objects by a threshold about the data along z axis in color frame in Kinect view. If

the coordinate along z axis object is higher than threshold, then it is defined as the robot hand; otherwise, it's the industrial component. The concept of multi-objects tracking is indicated.



6.2.3 Position calibration

From Kinect view, we can get the information about depth along z axis. However, true position is altered due to quantization level among pixels. The scale from color frame to depth map should be measured before tracking. The length between each pixel is proportional to the vertical distance between the targets and Kinect. This relationship is also the same as the content mentioned in Object Recognition.

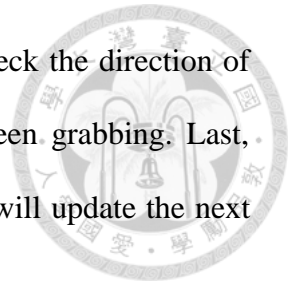
Once the tracking strategy is actuated, the static pose recognition will stop. The data include the position of the end-effector and object on the conveyor. The element will move straight forward through the conveyor. Thus, only the pose of elements will be updated and orientation maintain present. Then, the control center will convey new command to the manipulator for grasping tasks.

6.3 Modified Grasping

It is worth notice that the end-effector may obey the next message derived from tracking strategy. However, this has little help in grabbing articles. To make sure the completeness of whole grasping movement, the gripper should take aim-off into consideration while it carry out object tracking. Therefore, the main point turns out to be accurate grasping pose through tracking process. When some problems occur on the conveyor, the arm may not know without extra indication.

Thanks to the technique of eye-in-hand, the additional information will be added under visual feedback system. A webcam camera is allocated for modified grasping. When the robot receives instruction from control center to start tracking, the difference

among frames will be recorded through the process. First of all, check the direction of gripper. Next, webcam records the frame while the gripper has been grabbing. Last, revision including grasping pose and orientation will be saved and will update the next pose and orientation if the system recognizes the same object again.



6.3.1 Moving calibration

Every time the hand of robot arm start to execute moving task, the trajectory follow the commands derived from trajectory planning and eye-in-hand vision. Under the view of webcam, the edge of conveyor will emerge on the upper side of image. Because the moving path is linear on the conveyor, following the slope of the edge of conveyor is the way to correct the motion of robot.

To search a graph in an image, there is lots of image processing to select. One simple method is Hough transform which is well known as object detection. Under the background, the object is just a set of colorful point in camera view. Those point set will be mapped to a certain point or a high dimension plane. This can gain a parametric equation to represent all possible condition to describe the characteristic of pattern. Furthermore, search the extreme value to find out the position of the pattern. Last, collect all similar points together to illustrate the feature which is part of element.

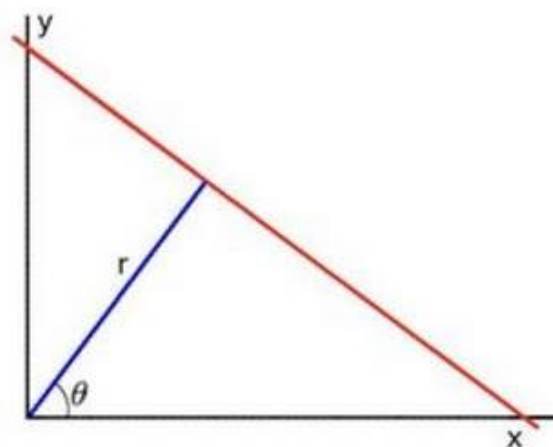
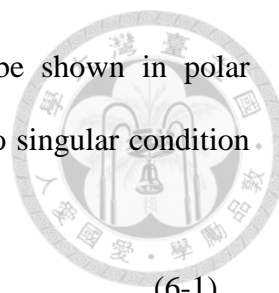


Fig. 6.3 Line description in polar coordinate.



To find out the line edge of conveyor, line equation can be shown in polar coordinate in Fig. 6.3. The advantage of polar parametric form is no singular condition under sinusoidal function domain.

$$r = x \cos \theta + y \sin \theta \quad (6-1)$$

Actually this condition simply contains a set of all possible solution of r, where the maximum count happens to. However, it takes lots of time in calculation. To decrease time consumption, the generalized of Hough transform helps to develop a structure to compute the candidate reference points. The implementation for general case can be written as:

$$x_t = (x_0 \cos \theta - y_0 \sin \theta) s \quad (6-2)$$

$$y_t = (x_0 \sin \theta + y_0 \cos \theta) s \quad (6-3)$$

where x_0, y_0 is initial point, θ is rotation angle and s is uniform scaling value. To imply the reference point, the equation turns out to be:

$$x_c = x - (x_0 \cos \theta - y_0 \sin \theta) s \quad (6-4)$$

$$y_c = y - (x_0 \sin \theta + y_0 \cos \theta) s \quad (6-5)$$

The generalized method didn't mean to create a common way to find any kind of shape but to record all edges into a table corresponding to a reference point. After the slope of edge is gained, the orientation of gripper will gradually change through moving process.

6.3.2 Pose modification

To check out whether the task is successful or not, the webcam records image while the gripper is holding the object. The conceptual structure is shown in Fig. 6.4. To find out any variation of grasping pose, webcam will first store lots of correct pose picture info offline database. After gathering the poses as a standard level, the mission starts to grasp moving target. In this step, the grasping pose may update its position and

orientation based on calibration in previous subsection and modified pose.

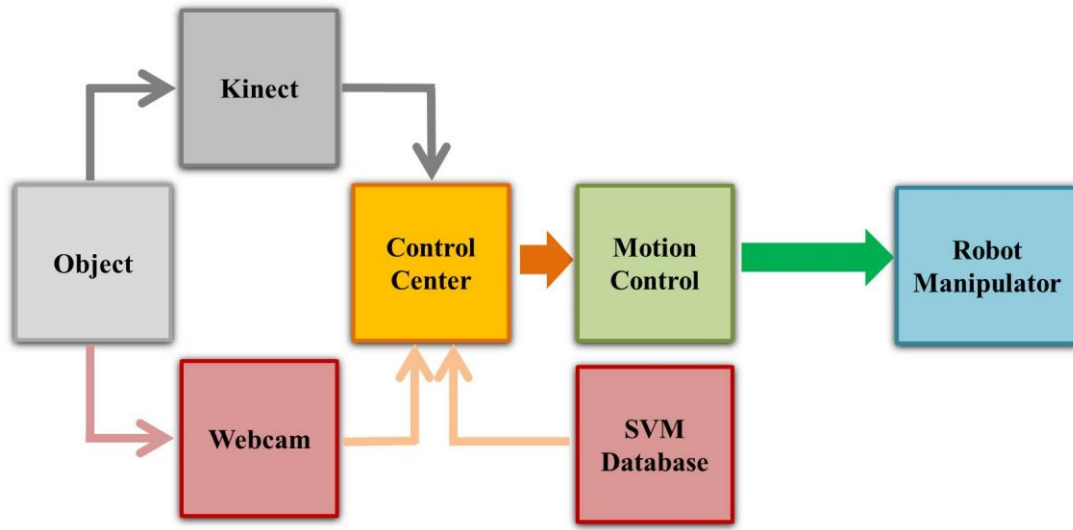


Fig. 6.4 System Structure for modified grasping.

First, an aim-off happens because there is relative motion between moving target and end-effector. Thus, an aim-off is defined as:

$$\Delta P = V_c \Delta t \tag{6-6}$$

where ΔP is aim-off of gripper to the moving target, V_c is the velocity of conveyor and Δt is sampling time, nearly 0.5 sec. Then, the next state of position can be calculated as:

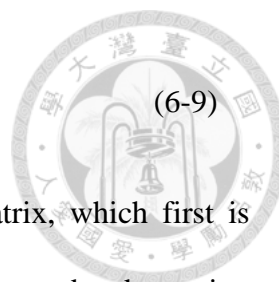
$$P_{t+1} = R_c R_t P_t + \Delta P \tag{6-7}$$

where R_c is the rotation matrix of calibration, R_t and P_t is previous state of rotation matrix and pose. The actual command to robot will be a 4×4 transformation matrix.

The general expression is

$$T_{t+1} = \begin{bmatrix} R_{t+1} & P_{t+1} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_c R_t & P_t \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} I_{3 \times 3} & \Delta P \\ 0 & 1 \end{bmatrix} \tag{6-8}$$

In addition, the revision will be recorded into offline database for the same type and pose next time. To clarify the state of position, the offline part collects the average value from time zero to time t. The equation shows as following:



$$P_n = \frac{1}{t} \left(\sum_{x=1}^t R_{c_avg} R_x P_x \right) \quad (6-9)$$

where R_{c_avg} means the average value of calibration rotation matrix, which first is regarded as identity matrix. After combing new grasping pose with the updated grasping modification, robot manipulator will work better in the next task.

6.3.3 SVM database

To deal with the error in grasping moving objects, there is an offline database to record the difference between theoretical value and actual one. At first, the recording error will be introduced in the error model. After analyzing the following error and calibration error, the compensation for the grasping pose will sent to the training model. When the offline database is well developed, the modified value will add to the commands for robot end-effector to finish the task successfully.

■ Error definition

The errors contain following error, calibration error and friction error. Usually the friction error happens in the rotation, including the conveyor, actuators in transmission mechanism. The effect of friction in conveyor moving can be diminished by lubricating the motors and bearing box. In manipulator mechanism, the movement should be successfully and smoothly completed by commands, which friction is small enough to ignore. But this problem also can be reduced by lubrication. Thus, the friction will not be mentioned in this subsection.

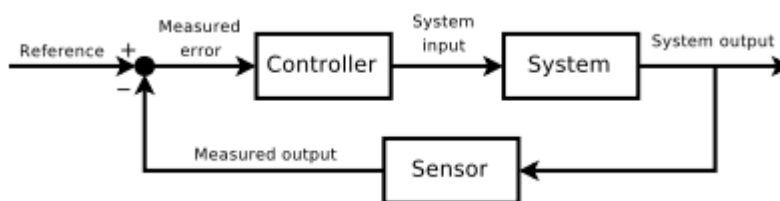


Fig. 6.5 A block diagram of a negative feedback control system



Fig. 6.6 Maxon motor control

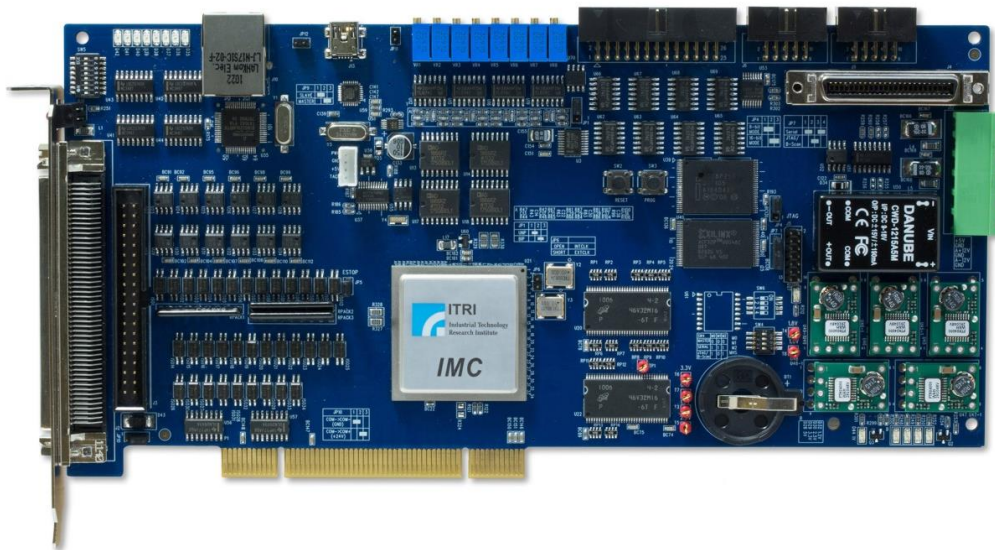
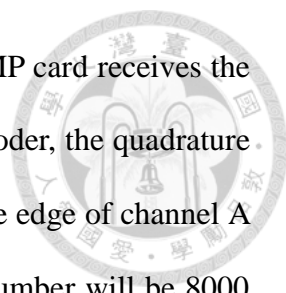


Fig. 6.7 Intelligent motion control platform

The following error is a kind of feedback signal to deal with robot control system. A block diagram is shown in Fig. 6.5. A simple feedback control is actuated by the error count. Besides, the signal always turns out to be a digital value to compare with the designated command. Motion control can be solved by the controller, which has already installed the software to adjust the parameter or monitor the state of motors. It is common in position control and velocity control to use following error to update the movement of robot. In the research, the motor driver is Maxon motor control, order number 145391, and the control part is ITRI intelligence motion control platform (ITRI IMP card). They are shown in Fig. 6.6 and Fig. 6.7.

Actually, there is a parameter for each joint driver to restrict the range of following



error. To enhance the precise accuracy in the feedback signal, the IMP card receives the edge of each PWM signal from the encoders. For an orthogonal encoder, the quadrature pulses will produce 4 edges, including the positive edge and negative edge of channel A and B. When the count in the encoder is 2000 per revolution, the number will be 8000 in the controller. Therefore, this kind of amplification enables to raise the resolution in calculation to sense the following condition. Commands from controller are sent to the encoders and motors. The feedback signal will transmit the state to the control center at next sampling time. To figure out the unexpected rising following error in the feedback control system, there are four cases to discuss.

Case 1: If there is a reverse orientation or contrary sign, the wiring may be open in somewhere. This usually happens in an old or frequent-used device. It is evident that the connector and adaptor should be replaced in the circuit.

Case 2: If the following error is large and the circuit remains regular, the PID parameters may not be suitable in the robot system. Actually, the PID parameters should be separated with proper value. They are helpful to control velocity or current of servo motor. It is essential to implement fine tuning in PID control to perform stable convergence of following error in robot manipulator control.

Case 3: If the motor moves counterclockwise when a clockwise rotating command is given, the phase of analog output in the encoder connects in the opposite side. Actually the connection is inverse polarity, it is common to change the sign of polar parameter in the controller or just exchange connection.

Case 4: If there is a high speed movement for manipulator, the high velocity or accelerated command usually causes huge following error. Due to the endurance of hardware structure, the excessive friction between gear box and mechanism lead to exhaustive motor rotation and component displacement. The possible method is to

doi:10.6342/NTU201703382

transmit relative low velocity or acceleration to the robot or make the mechanism solidier enough to support this condition.

The following error can be monitored by controller in the feedback control system. Therefore, the error model will mainly be the calibration deviation.

■ Calibration deviation

The calibration deviation is a kind of mechanical position error experienced in setting a positioning device to align with a desired reference frame. The calibration error in a measurement indicates how well the measuring instrument has been made, and is usually quoted by the manufacturer. It is usually quoted as a percentage of the reading or a percentage of full scale deflection. The error is the difference between the desired reference frame and the set reference frame. Error contributors may include mechanical tolerances, gravity deflection, backlash, electrical noise, and control system errors. In this case, it is a kind of artificial error in camera calibration.

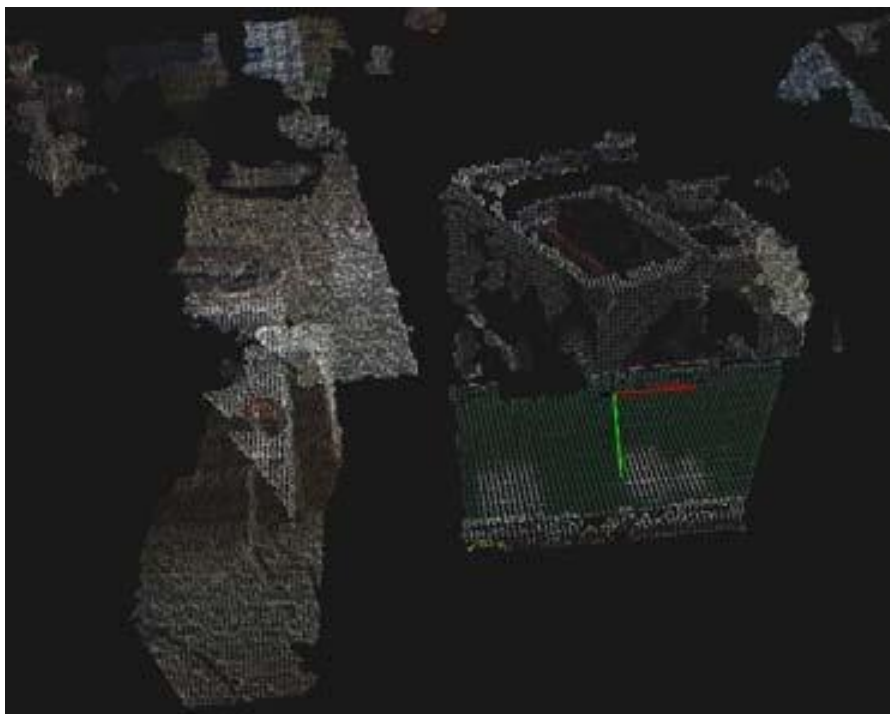


Fig. 6.8 The artificial calibration illustration

The relationship among robots and sensors and other supporting device are helpful for manipulator to complete pick and place task. The relative coordinate has already been derived from matrix operation in previous section. However, there are still some errors in the moving process after first calibration. In the previous subsection mentioning calibration, it is mainly about the reprojection error. The reprojection error is a geometric error corresponding to the image distance between a projected point and a measured one. It is used to quantify how closely an estimate of a 3D point recreates the point's true projection. This has been done with intrinsic matrix and extrinsic matrix. However, the relationship between Kinect and robot manipulator is artificial calibration. This absolutely faces errors.

In Fig. 6.8, this is PCL view from Kinect sensor. There is a coordinate with three colors corresponding to three coordinate: red is x-axis, green is y-axis and blue is z-axis. This symbol helps to understand the Kinect coordinate distribution. By using keyboard control, the coordinate will match with the left front corner of robot base. This can only be done by naked eye. The resolution of Kinect is low and the view with long distance is fragmented. Therefore, it is possible to carry out huge error.

The calibration deviation can be separated into two parts: translation and rotation. The translation error is

$$T_t = T_e = T_x + T_y + T_z \quad (6-10)$$

In the rotation part, the end-effector pose owns three rotation matrices by an angle θ about the x-axis, y-axis and z-axis. For column vectors, each of these basic vector rotations appears counterclockwise when the axis about which they occurs points toward the observer, the coordinate system is right-handed, and the angle θ is positive. Along each axis, the product of rotation matrix can be written as



$$\begin{aligned}
 R_{rot} &= R_{roll}R_{pitch}R_{yaw} \\
 R_{roll} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \\
 R_{pitch} &= \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \\
 R_{yaw} &= \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}
 \end{aligned} \tag{6-11}$$

Therefore, the total rotation error is

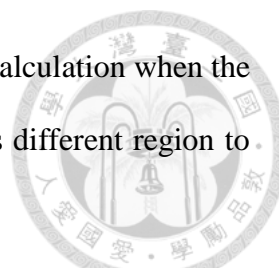
$$T_r = R_e^{\Delta\theta} T_i = R_{yaw}R_{pitch}R_{roll}T_i \cong R_{yaw}T_i \tag{6-12}$$

In the experiment, the magnitude of difference in roll and pitch is negligible. The error can be simplified into yaw angle. Therefore, the calibration deviation can be taken into consideration.

Support vector machine

Support Vector Machines (SVM) is one of famous classification algorithms [37]. It can be traced back from Vapnik, who put forward a statistics learning theory as a new machining learning method. SVM has great superiority on numerous cases, such as small samples, non-linear model and high dimension analysis. It is well performed in words recognition, handwriting recognition, three dimension object recognition, human face recognition and patterns classification. To develop high learning ability in the SVM model, the decision for limited training set will be trained and receives less error for individual testing set.

The concept of SVM is a hyperplane as separation for different sets. Actually, the problems are often high dimension data, which cannot easily describe in the figure. For example, in two dimension plane, we hope to separate the black and white points as much as possible in Fig. 6.9. We hope the borderline to be clear and the bigger margin



for precise recognition. Otherwise, it is possible to produce error in calculation when the accuracy is not credible enough. In this case, different sign symbols different region to create a classifier.

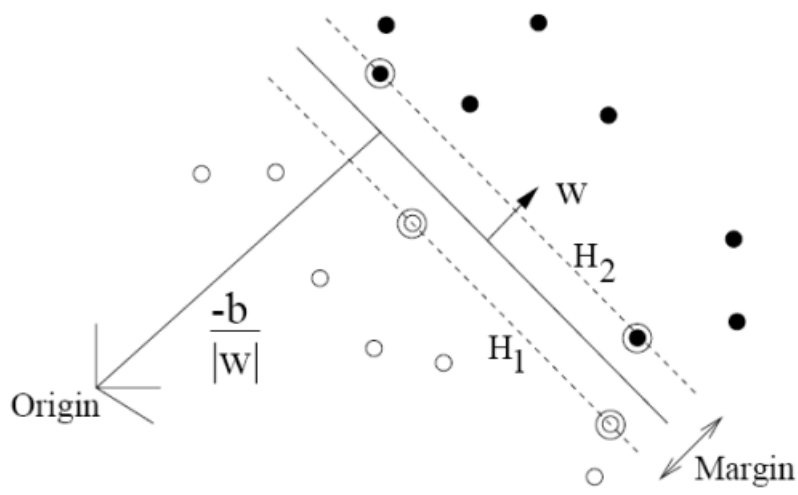


Fig. 6.9 Two dimension example in SVM analysis

This plane is called separating hyperplane. The plane with the biggest margin is called optimal separating hyperplane. To find out the optimal one, we have to find out the support hyperplane. When we want to make sure the quantity of points near the margin, there are some restrictions to find out match condition.

$$L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^N \alpha_i [y_i(w^T x_i - b) - 1] \quad (6-13)$$

The polar value will reveal when

$$y_i(w^T x_i - b) - 1 = 0, \alpha_i \geq 0 \quad (6-14)$$

$$y_i(w^T x_i - b) - 1 > 0, \alpha_i = 0 \quad (6-15)$$

These points are just located in the support hyperplane. Therefore, they are called support vector and their α_i will remain positive. Next, we recognize the type of point to corresponding set.

Primarily, the SVM method is a kind of binary classifier. But multiple

classifications are in fact common in most cases. We need to use some strategy to deal with multiple sets.

1. One-against-rest: If there are totally k classes, then we create k SVM classifiers. The m^{th} SVM can separate the m^{th} elements from others. That is to say, each classifier only recognizes one class of elements.
2. One-against-one: To any two of all points, we create a specialized SVM for them. If there are k points to separate, we need $k(k+1)/2$ SVM classifier. The fundamental function can only separate two classes in a group. If there are more than one category, it is still unable to classify extra class.

Therefore, we use SVM algorithm to recognize the picture is straight or tilt. Next, we start to describe the training model.

Training model

First, I start to define the straight pose and tilt one. In this research, there are three types of industrial elements with different shape, size and stable pose. The numbers of picture for each element is 240. The SVM can only separate two categories. However, there are three conditions to describe the problems: correct pose, skewed pose with clockwise or counterclockwise type. To analyze one type of element, we should provide three SVM model to classify the poses. When those pictures are taken, we will do HOG on each image. The histogram of oriented gradients (HOG) is a feature descriptor used in computer vision and image processing for the purpose of object detection. This technique counts occurrences of gradient orientation in localized portions of an image . We use this method to calculate the edge orientation histograms of features and shape contexts. The input frames will be resized to 208 x 416 to do HOG.

Next, all HOG files will be given a label whether it is positive data or negative data. All the steps are ready for SVM method. The training machine is Lib-SVM. So, the

doi:10.6342/NTU201703382

input files should be translated into Lib-SVM type. Separating the action into positive pose and negative one is setting up the training model. In a model, only one pose will be assigned positive flag while others are all negative. The data will be shown in the following figures.

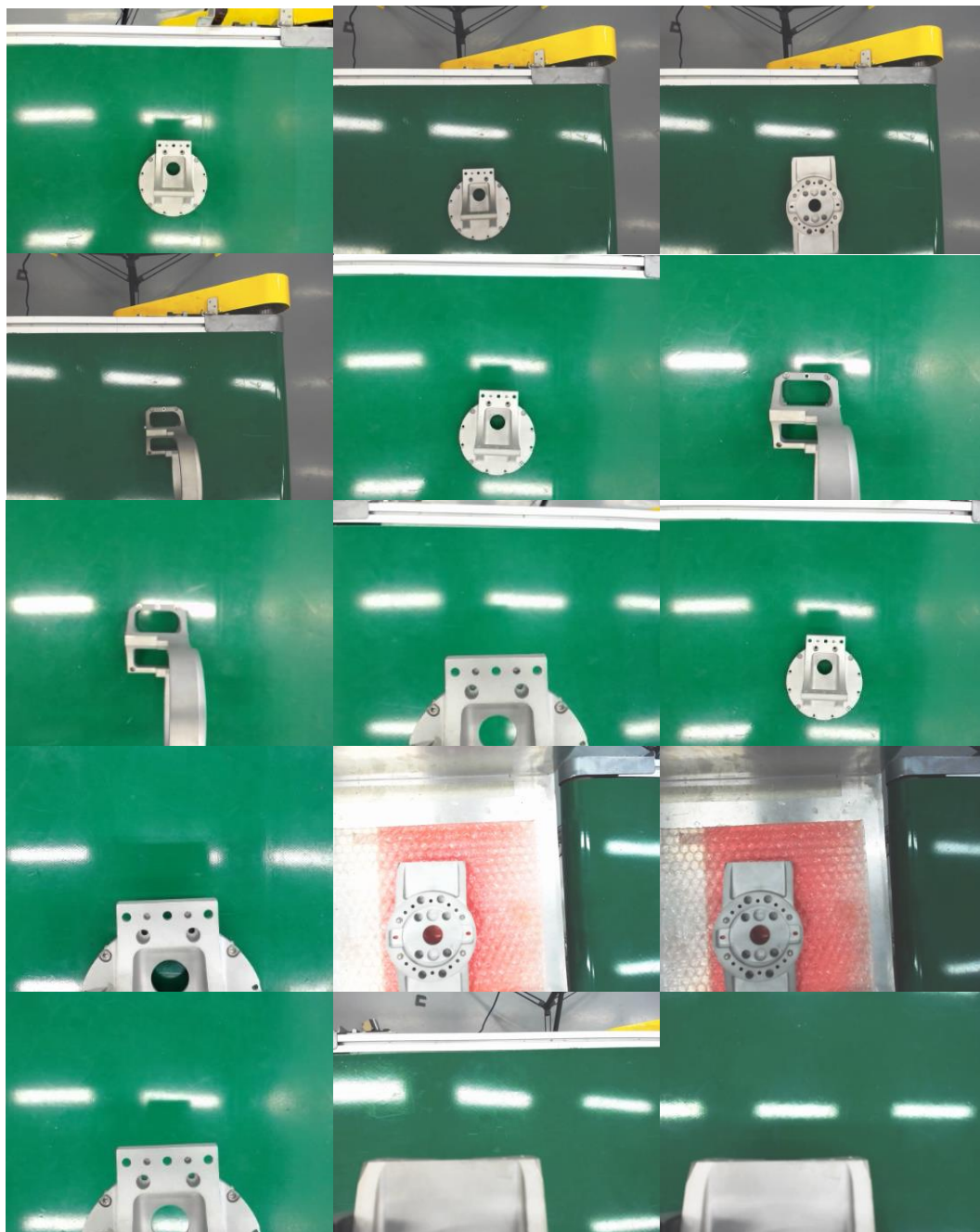
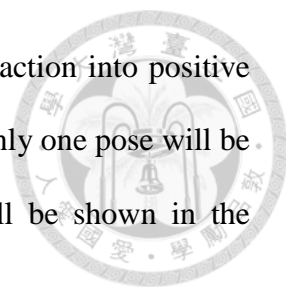


Fig. 6.10 The correct pose of the element.

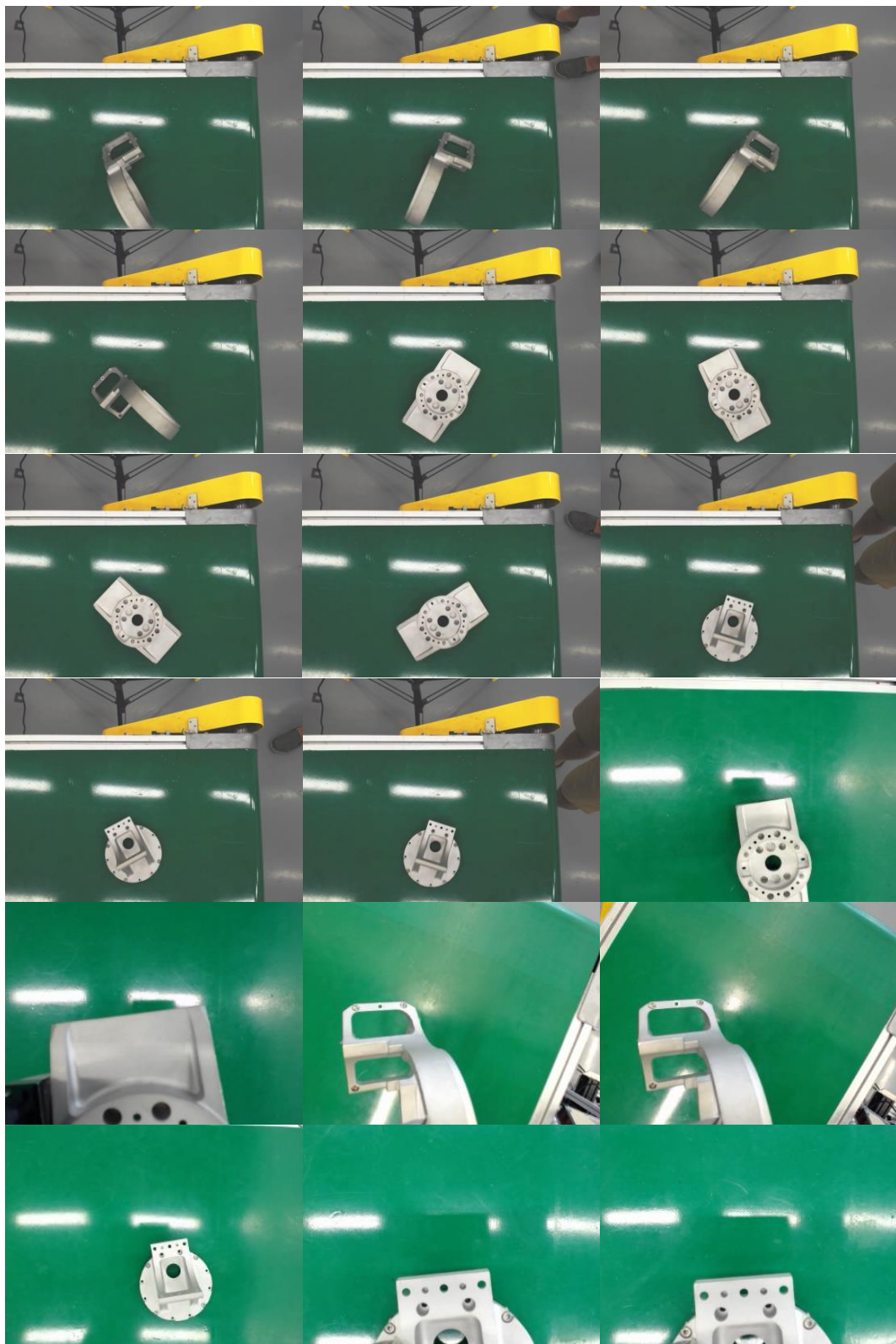


Fig. 6.11 The skewed pose of the element.

Chapter 7 Experimental Results and Discussion

There are two steps to carry out in the complicated scenario. The environment is built in Fig. 9. There are one robot manipulator, a conveyor, two platforms, two vision sensors and three objects.



Fig. 7.1 Experimental Scenario

7.1 Object Recognition and Fetching

The object recognition and fetching system is the main technique because the object information in tracking has been predefined in object recognition. Besides, without a high recognition rate, the operation database will not be able to find out a correct grasping pose to finish pick and place task. Therefore, we start to describe the process of object recognition results. Every object should be placed in the working area shown in Fig. 7.2. Only when the object comes inside the working area do the Kinect

doi:10.6342/NTU201703382

start to do object recognition and transmit to the control center.

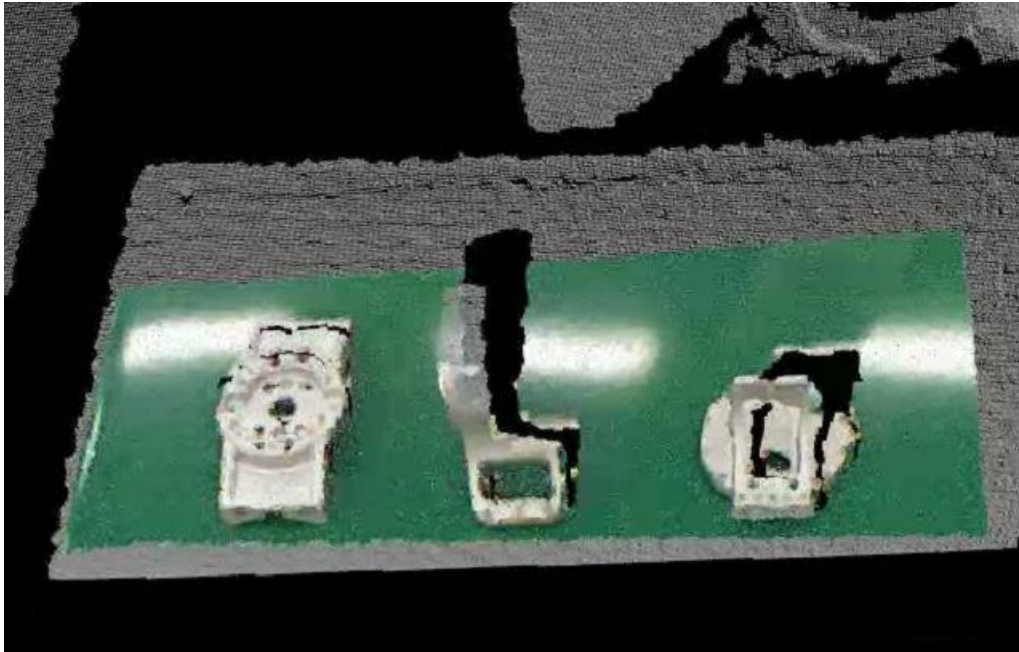
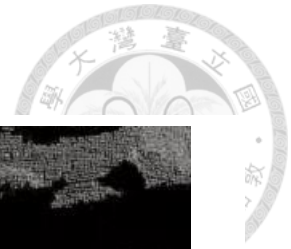


Fig. 7.2 The working area of object recognition system

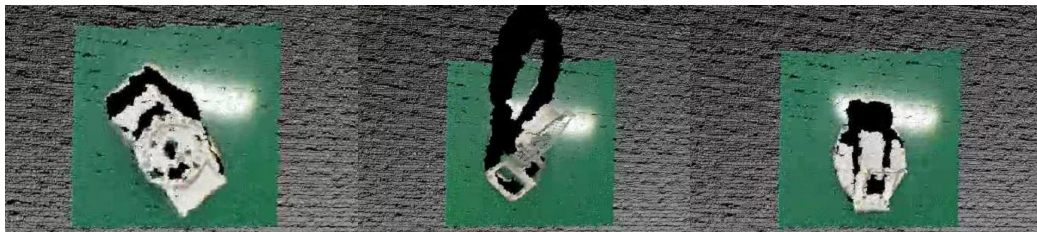


Fig. 7.3 The ROI of each object

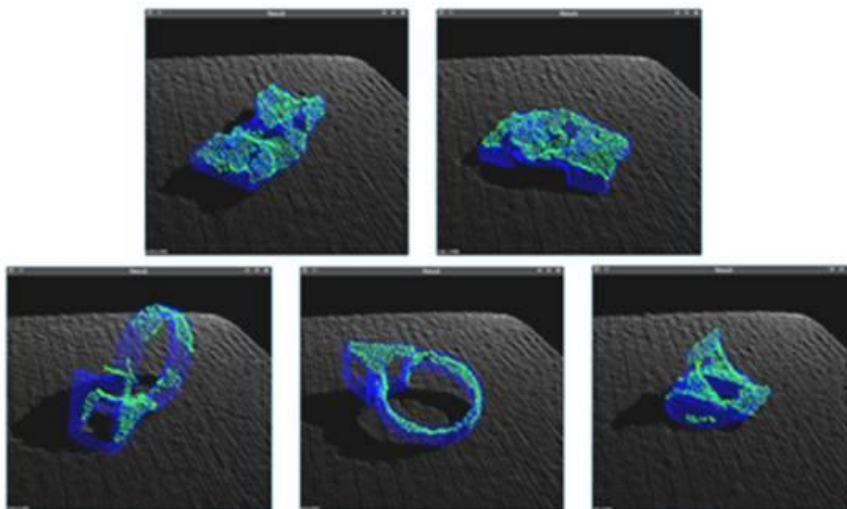


Fig. 7.4 Object recognition results

In Fig. 7.3, each element owns its region of interest to be segmented in Kinect view. The results of object recognition in PCL are shown in Fig. 7.4. The blue dots are CAD model points while the green ones are the cluster points. The more green dots replace the blue ones, the higher success rate a recognition system owns.



Fig. 7.5 Object fetching results

The results of object fetching are shown in Fig. 7.5. the taught grasps are successfully implemented in the experiment. It can be seen from the figure that we only need to reach the robot one grasp for each group of stable poses and the operation database is able to select a correct grasp and transform the grasp based on the object recognition result. The robot manipulator can then follow the given command to pick up the object successfully.

7.2 Object Tracking

The main goal in tracking is to grasp moving object on the conveyor before the target falls down. We first simulate the process in the MATLAB, shown in Fig. 7.6.

doi:10.6342/NTU201703382

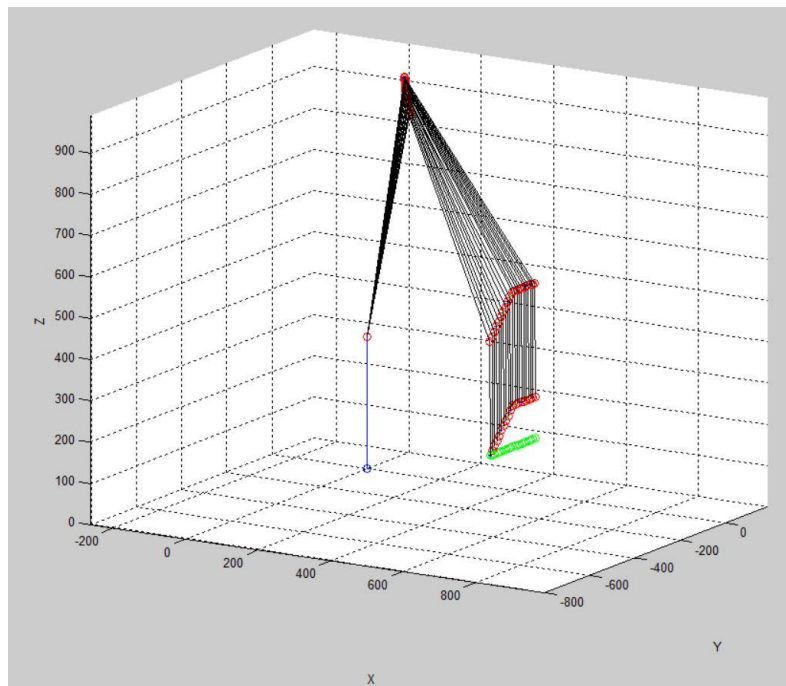


Fig. 7.6 Simulation of tracking an object

The basic concept in tracking is to follow the object and grasp later. Before the object reach to the terminal edge, the manipulator track the target with updated data from visual sensors. There is a threshold value in the y-axis. If the value of y-axis is higher than threshold value, the arm will move in high-speed to hold the object with a predicted compensation. This predicted compensation is associated with some parameters, including the velocity of robot arm (v_{arm}) and conveyor (v_{cnvr}), and the ratio of two velocities (v_{arm}/v_{cnvr}). Actually, if v_{cnvr} is too fast, the manipulator is unable to track the target in any magnitude of velocity. Later, two methods will be shown in the following figures. The first part is eye to hand.

Vision sensors have huge effect on the performance of conveyor tracking. The first part is object recognition in Kinect sensing. The process of eye to hand is shown in Fig. 7.7. When the type and pose of object is obtained, Kinect change the view to tracking strategy. Rather than static object detection by recognition algorithm, the moving element puts much more unexpected factors during grasping process.

doi:10.6342/NTU201703382

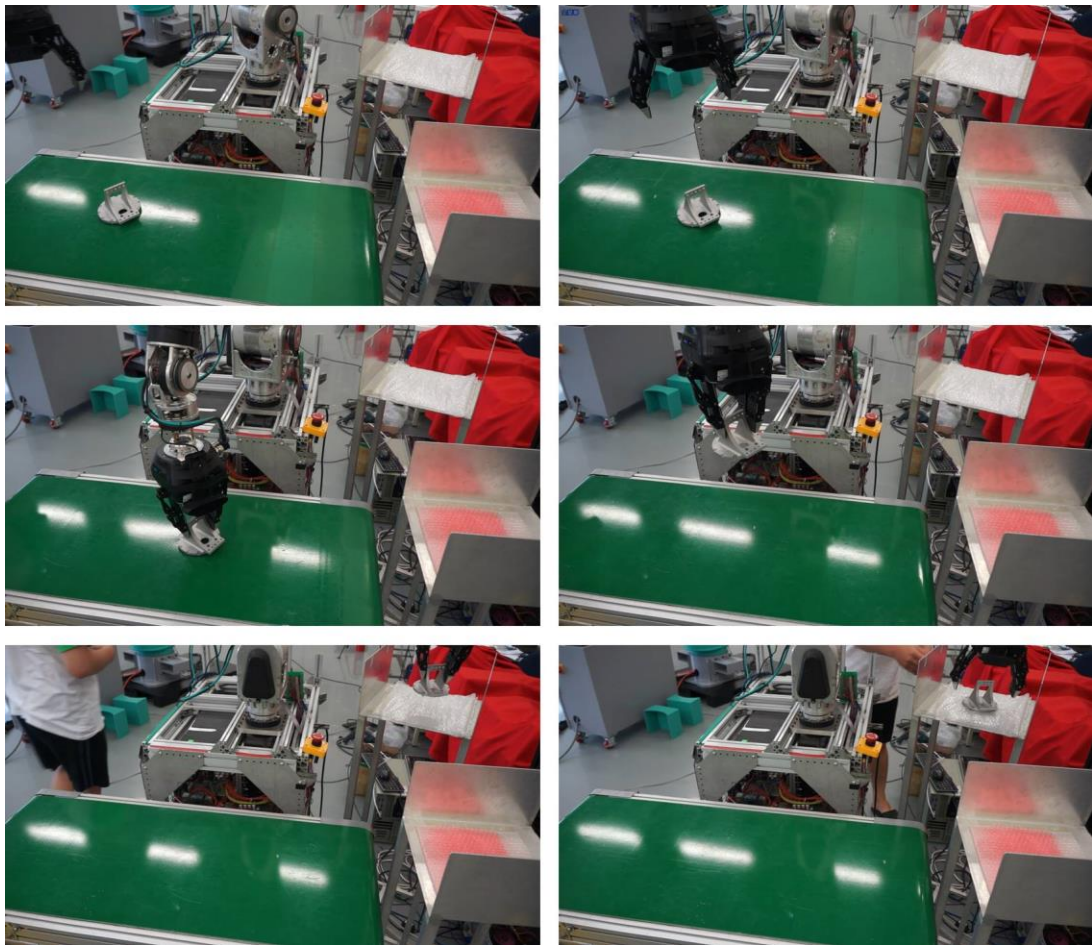


Fig. 7.7 The process of eye to hand tracking

The arm will go to the pre-pick point first. When Kinect sends the position and orientation of object, the robot will start to predict final position of target and reach to the object directly above. Then, the end-effector goes down to grasp it in taught pose. Finally, the object is placed on the designated platform. This process is common in parallel robot tracking.

The second skill is tracking moving target by webcam camera on the conveyor shown in Fig. 7.8. When a static element finishes recognition and the result is correct, the conveyor starts to send the object. By means of webcam, the instant pose and orientation of component will be captured to let the gripper know where the target is right now.

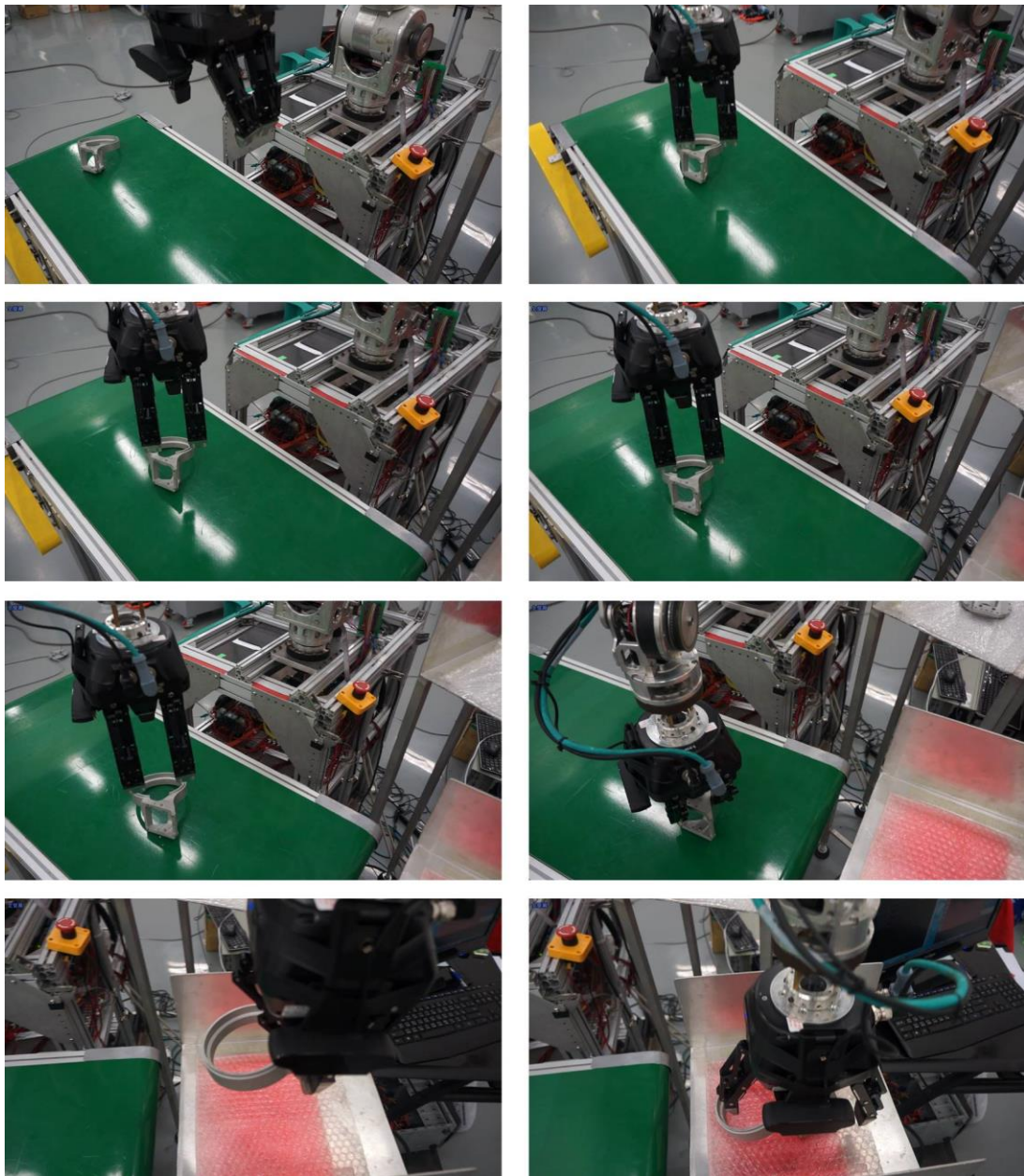
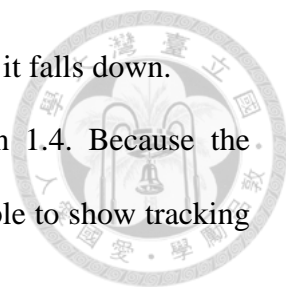


Fig. 7.8 The process of eye in hand tracking.

In this method, there is no pre-pick pose for robot manipulator. The first sight of object is still derived from object recognition system because robot should know the object type and the corresponding grasping pose. Next, the arm will move overhead relative to the target. There will be several times of tracking, which are decided by the object of y-axis value. This figure presents three times on tracking and run to the target to hold. Although it may takes time to update those desired information, the end-effector

doi:10.6342/NTU201703382



follow commands from the control center and catch the target before it falls down.

The velocity ratio of arm to conveyor should be larger than 1.4. Because the distance on the conveyor is only 1 meter and only half distance is able to show tracking action, the velocity of conveyor may not pass objects too fast.

7.3 Modified Grasping

7.3.1 Object pose modification

The other technique is grasping modification as shown in Fig. 7.9. First, the offline database is obligated to gather the correct pose of grabbing. Before capturing the target, a picture will be photoed to check out the edge of conveyor is altered or not during tracking process. If the orientation changes, it adjusts the gripper to rotate exact direction.

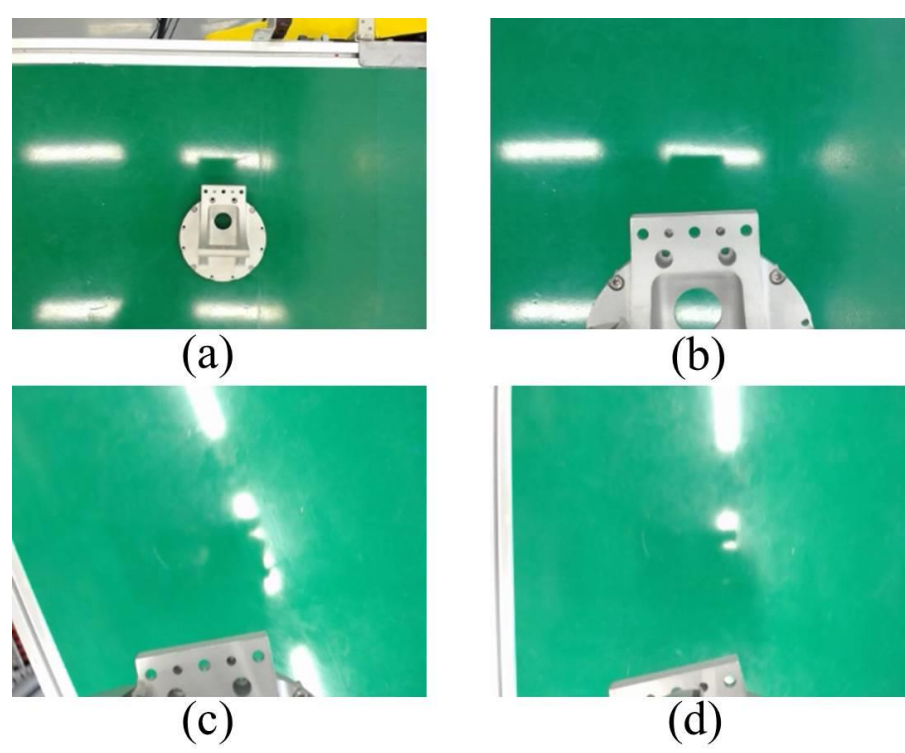
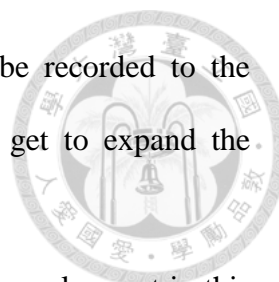


Fig. 7.9 The conveyor edge detection. Correct grasping in top row and incorrect grasping in bottom row.



Later, end-effector is seizing the object. Correct poses will be recorded to the database. When the task begins, those incorrect photos will also get to expand the feasibility of eye-in-hand system.

After two experiments are implemented, it is obvious that the complex part in this research is useful. Moreover, time consumption in the recognition and tracking is not bad. Based on the visual feedback system, the image data have been defined as necessary modification, which assists robot manipulator to analyze the current state of object.

7.3.2 Classifier results

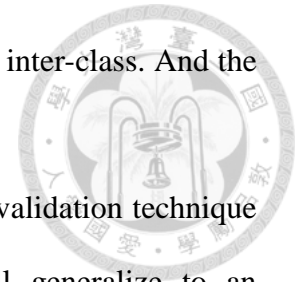
In previous subsection 6.3.3, there is an algorithm SVM model for training object poses. After create the data set, we start to train the Lib-SVM model. There will be at least three models for each type. At first, a single-class Lib-SVM is used for one of the elements. 1 symbols positive and -1 symbols negative. The probability results are listed in Table 7-1. It is obvious that the models are suitable for each type because they are based on static poses in each group. However, they provide so many models to do only two categories in single-class.

Probability	Type 1	Type 2	Type 3
Correct	94.6%	95%	95.83%
Skewed	99.17%	98%	99.17%

Table 7-1 Single-class Lib-SVM recognition result

For better speed and performance in running, we use Lib-SVM multi-class to set up database containing three poses: correct, clockwise skewed and counterclockwise skewed. -1 symbols negative, and 1~3 match with pose 1~3, which means there are three poses in the model. Besides, to enhance the model with high accuracy, two

evaluations are in need for testing. One is intra-class, and another is inter-class. And the method I used is leave-one-out cross-validation [39][40].



Leave-one-out cross-validation: Cross-validation is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. In a prediction problem, a model is usually given a dataset of known data on which training is run (training dataset), and a dataset of unknown data (or first seen data) against which the model is tested (testing dataset). The goal of cross validation is to define a dataset to "test" the model in the training phase (the validation dataset), in order to limit problems like overfitting, give an insight on how the model will generalize to an independent dataset (an unknown dataset), etc. Leave one out means the unknown dataset is also part of original dataset. It will be regarded as unknown data while put into training set in other cases. This dataset is not truly testing set. For n groups of data, choose one group from all for a test and others are trained to be a model. Then, test the model by that one group to calculate the success rate. It will be done n times, and finally average all models' rate as the result of these data.

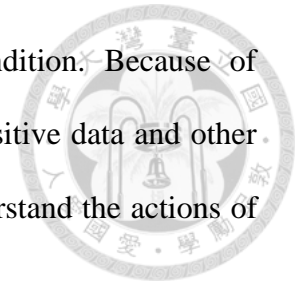
Intra-class: There are n poses for each object. One of the poses tests the model trained by others. Use leave-one-out cross-validation and record the rate of each type.

Inter-class: Take one of the elements for testing, and others do the training. The method is also leave-one-out cross-validation.

To judge the performance of the models, it is important to calculate accuracy and precision. In this research model, accuracy is used as a statistical measure of how well a

doi:10.6342/NTU201703382

binary classification test correctly identifies or excludes a condition. Because of multi-class in binary classification, we classify three poses into positive data and other poses into negative data. Among the test, count four values to understand the actions of model.



True positive (TP): Positive data are correctly recognized

True negative (TN): Negative data are correctly recognized

False positive (FP): Negative data are regarded as positive data

False negative (FN, Deletion error): Positive data are regarded as negative data

There are three errors possibly existed during the test.

Insertion error: The same as false positive

Deletion error: The same as false negative

Substitution error: Error exists from value classified into TP but not assigned to true value because positive values contain three poses in multi-class

In the statistics, calculate the rate showing below to figure out how well the models behave. In pattern recognition, information retrieval and binary classification, precision is the fraction of relevant instances among the retrieved instances, while recall is the fraction of relevant instances that have been retrieved over total relevant instances in the image. Both precision and recall are therefore based on an understanding and measure of relevance. Here are the equations.

$$\text{Precision (True Positive Rate or Sensitivity)} = \frac{T_p}{T_p + F_p}$$

$$\text{Recall (Positive predictive value)} = \frac{T_p}{T_p + F_n}$$

$$\text{True negative rate (Specificity)} = \frac{T_n}{T_n + F_p}$$

$$\text{Substitution error rate} = \frac{N_s}{T_p}$$

$$\text{Accuracy} = \frac{T_p + T_n}{T_p + T_n + F_p + F_n}$$

The intra-class results are listed in Table 7-2, and the inter-class is in Table 7-3.



Type	0	1	2
TP	207	209	203
TN	29	26	29
FP	3	4	1
FN	3	1	7
Substitution error (NS)	0	0	3
Total	242	240	240
Precision	98.57%	98.12%	99.51%
Recall	98.57%	99.52%	96.67%
True negative rate	90.63%	86.67%	96.67%
Substitution error rate	0%	0%	1.48%
Accuracy	97.52%	97.92%	96.67%

Table 7-2 Intra-class statistics

Type	0	1	2
TP	208	210	209
TN	26	26	30
FP	6	4	0
FN	2	0	1
Substitution error (NS)	2	0	1
Total	242	240	240
Precision	97.19%	98.13%	100%
Recall	99.05%	100%	99.52%
True negative rate	81.25%	86.67%	100%
Substitution error rate	1%	0%	0.5%
Accuracy	96.7%	98.33%	99.58%

Table 7-3 Inter-class statistics

Chapter 8 Conclusions, Contributions and Future Works

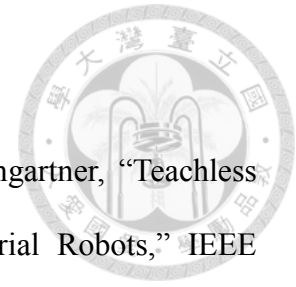


This thesis presents a strategy to provide dynamic information to robot arm under sophisticated environment. The contribution of this research can be summarized as follows:


1. Propose a specialized architecture for a sensor-integrated object tracking system including eye to hand and eye in hand system
2. Propose a modified architecture to check the variation in grasping pose including translation and orientation

By means of object fetching system, the robot manipulator is able to identify and pick up the recognized object on the conveyor. With the help of vision sensors, the dynamic factor in the moving object can be captured and transmitted to the robot manipulator for grasping. It seems that the results are well demonstrated. However, in grasping modification, it may face some restriction under the size and type of elements. To revise the pose and orientation, some component of object should go inside the webcam view so that the new pose will be sent. Besides, the integration of vision sensors faces the consumption on programming, which spends much time updating image data from Kinect tracking and webcam fetching. In view of this, the structure can be developed by more precise resolution but low time consuming vision sensors. Moreover, further improvement in grasping issue is also integrated with various sensors in the environment. This may be a functional application in packaging in the production line.

REFERENCE




- [1] M. Perrollaz, S. Khorbotly, A. Cool, J. Yoder and E. Baumgartner, "Teachless teach-repeat: Toward Vision-based Programming of Industrial Robots," IEEE International Conference on Robotics and Automation, pp.409 - 414, 2012.
- [2] Man vs machine table tennis show down that went viral: <http://www.dailymail.co.uk/sciencetech/article-2578633/The-Man-vs-Machine-table-tennis-showdown-went-viral-actual-cleverly-shot-TV-ad.html>
- [3] L. Zhang, S. Lyu, and J. Trinkle, "A Dynamic Bayesian Approach to Real-Time Estimation and Filtering in Grasp Acquisition," in IEEE International Conference on Robotics and Automation (ICRA), may 2013, pp.85-92.
- [4] D. Katz, M. Kazemi, J. A. Bagnell and A. Stentz, "Clearing a Pile of Unknown Objects using Interactive Perception," in IEEE International Conference on Robotics and Automation (ICRA), may 2013, pp.154-161.
- [5] N. Chavan Dafle, A. Rodriguez, R. Paolini, B. Tang, S. S. Srinivasa, M. Erdmann, M. T. Mason, I. Lundberg, H. Staab and T. Fuhlbrigge, " in IEEE International Conference on Robotics and Automation (ICRA), may 2014, pp.1578-1585.
- [6] N. A. Vien and M. Toussaint, " POMDP Manipulation via Trajectory Optimization," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 242-249, 2015.
- [7] A. Ajoudani, C. Fang, N. G. Tsagarakis, and A. Bicchi, " A Reduced-Complexity Description of Arm Endpoint Stiffness with Applications to Teleimpedance Control," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4636 - 4641, 2013.

- 
- [8] Y. Suzuki, K. Koyama, A. Ming and M. Shimojo, " Grasping Strategy for Moving Object using Net-Structure Proximity Sensor and Vision Sensor," in IEEE International Conference on Robotics and Automation (ICRA), may 2015, pp.1403-1409.
- [9] M. Mukadam, X. Yan, and B. Boots, " Gaussian Process Motion Planning," in IEEE International Conference on Robotics and Automation (ICRA), may 2016, pp.9-15.
- [10] J. H. and D. D. Lee, " Learning High-Dimensional Mixture Models for Fast Collision Detection in Rapidly-Exploring Random Trees," in IEEE International Conference on Robotics and Automation (ICRA), may 2016, pp.63-69.
- [11] K. Shimonomura, H. Nakashima, and K. Nozu, " Robotic grasp control with high-resolution combined tactile and proximity sensing," in IEEE International Conference on Robotics and Automation (ICRA), may 2016, pp.138-143.
- [12] M. Ewerton, G. Maeda, G. Neumann, V. Kisner, G. Kollegger, J. Wiemeyer and J. Peters, " Movement Primitives with Multiple Phase Parameters," in IEEE International Conference on Robotics and Automation (ICRA), may 2016, pp.201-206.
- [13] N. Wakisaka, R. Kikuuwe and T. Sugihara, " Fast Forward Dynamics Simulation of Robot Manipulators with Highly Frictional Gears," in IEEE International Conference on Robotics and Automation (ICRA), may 2016, pp.2096-2101.
- [14] Y. H. Tsai, "7-dof redundant robot manipulator with multimodal intuitive teach and play system," *Master's Thesis, Electrical Engineering, National Taiwan University*, 2014.
- [15] R. S. Hartenberg and J. Denavit, *Kinematic synthesis of linkages*: McGraw-Hill, 1964.



- [16] Robotiq gripper 3-finger gripper is available at
<http://robotiq.com/products/industrial-robot-hand/>
- [17] PISO-DA8U is available at
http://www.icpdas.com/root/product/solutions/pc_based_io_board/pci/pio-da4.html
- [18] PISO-Encoder600 is available at
http://www.icpdas.com/root/product/solutions/pc_based_io_board/motion_control_boards/piso_encoder600u.html
- [19] G. Grunwald, G. Schreiber, A. Albu-Schäffer, and G. Hirzinger, "Programming by touch: the different way of human-robot interaction," *Industrial Electronics, IEEE Transactions on*, vol. 50, pp. 659-666, 2003.
- [20] F. Libera, T. Minato, I. Fasel, H. Ishiguro, E. Menegatti, and E. Pagello, "Teaching by touching: an intuitive method for development of humanoid robot motions," in *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*, 2007, pp. 352-359.
- [21] Z. Li, L. Z. Ma, et al., "Rational quadratic B-spline curves with monotone curvature," *Journal of Information and Computational Science*, vol. 4, pp. 119-127, 2007.
- [22] J. Yongqiao, et al., "Research on consecutive micro-line interpolation algorithm with local cubic B-spline fitting for high speed machining," in *International Conference on Mechatronics and Automation (ICMA '10)*, 2010, pp. 1675-1680.
- [23] Q. G. Zhang, et al., "Development and implementation of a NURBS curve motion interpolator," *Robotics and Computer-Integrated Manufacturing*, vol. 14, pp. 27-36, 1998.

- 
- [24] Torsten Kroger, Adam Tomiczek, and Friedrich M. Wahl, "Towards On-Line Trajectory Computation," IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, October 9-15, 2006.
- [25] Torsten Kroger, and Friedrich M. Wahl, "Online Trajectory Generation: Basic Concepts for Instantaneous Reactions to Unforeseen Events," IEEE Transactions on Robotics, Vol. 26, No. 1, February 2010.
- [26] Torsten Kroger, and Jose Padiol, "Simple and Robust Visual Servo Control of Robot Arms Using an On-Line Trajectory Generator," IEEE International Conference on Robotics and Automation, RiverCentre, Saint Paul, Minnesota, USA, May 14-18, 2012.
- [27] Fabrizio Flacco, Torsten Kroger, Alessandro De Luca, Oussama Khatib, "A Depth Space Approach to Human-Robot Collision Avoidance," IEEE International Conference on Robotics and Automation, RiverCentre, Saint Paul, Minnesota, USA, May 14-18, 2012.
- [28] PointCloud Library (PCL) is available at <http://pointclouds.org/>
- [29] R. C. Luo and C. W. Kuo, "A Scalable Modular Architecture of 3D Object Acquisition for Manufacturing Automation," in IEEE 13th International Conference of Industrial Informatics (INDIN), 2015, pp.269-274.
- [30] W. Wohlking, A. Aldoma, R. B. Rusu, and M. Vincze, "3dnet: Large-scale object class recognition from cad models," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012, pp. 5384-5391.
- [31] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R. B. Rusu, *et al.*, "CAD-model recognition and 6DOF pose estimation using 3D cues," in *Computer*

Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on,
2011, pp. 585-592.



- [32] OpenGL is available at <https://www.opengl.org>
- [33] D. Shreiner and B. T. K. O. A. W. Group, *OpenGL programming guide: the official guide to learning OpenGL, versions 3.0 and 3.1*: Pearson Education, 2009.
- [34] Microsoft Corporation, 1 Microsoft Way, Redmond, WA 98052-7329, USA,
“Microsoft kinect homepage. <http://xbox.com/Kinect> (Mar. 28, 2011), Internet,
2011.
- [35] Zhang, Z. "A Flexible New Technique for Camera Calibration." IEEE
Transactions on Pattern Analysis and Machine Intelligence. Vol. 22, No. 11, 2000,
pp. 1330–1334.
- [36] R.B. Rusu and S.Cousins, " 3D is here: Point Cloud Library (PCL)", IEEE
International Conference on Robotics and Automation, pp.1-4, 2011.
- [37] SVM introduction is at
<http://www.cmlab.csie.ntu.edu.tw/~cyy/learning/tutorials/SVM2.pdf>
- [38] N. Dalal and B. Triggs, " Histograms of Oriented Gradients for Human Detection",
IEEE Computer Society Conference on Computer Vision and Pattern Recognition,
pp.1-8, 2005.
- [39] M. Dong, N. Wang, “Adaptive network-based fuzzy inference system with
leave-one-out cross-validation approach for prediction of surface roughness,”
Applied Mathematical Modelling, Vol. 35, Issue 3, pp.1024-1035, March 2011.
- [40] A. Vehtari, A. Gelman, J. Gabry “Practical Bayesian model evaluation using
leave-one-out cross-validation and WAIC,” Statistics and Computing, Vol. 27,
doi:10.6342/NTU201703382

Issue 5, pp.1413-1432, Sep. 2017.



VITA



姓名：廖俊豪

性別：男

生日：民國 80 年 6 月 26 日 (1991/06/26)

籍貫：中華民國 臺中市

學歷：

1. 民國 106 年 國立台灣大學電機工程學研究所畢業
2. 民國 102 年 國立台灣大學電機工程學系畢業
3. 民國 98 年 國立台中第一高級中學畢業

發表著作：

1. Ren C. Luo, Chun-Hao Liao, “Robotics Reconfigurable Conveyor Tracking and Dynamic Object Fetching For Industrial Automation”, accepted by 2017 IEEE International Conference on Industrial Informatics (INDIN 2017), Emden, Germany, July 24-26, 2017 (EI)
2. Ren C. Luo, Chun-Hao Liao and Mong-Hsun Kuo, “Non-Contact Collision Avoidance with Sensory Servo Control in Real Time for Industrial Automation”, accepted by 2017 IEEE Smart World Congress (SWC 2017), San Francisco, Bay Area, the United States, August 4-8, 2017 (EI)

榮譽事蹟：

- 2016 年 「2016 年智慧機器人創意競賽 國產工業機器人組」 榮獲團體組 **季軍**
- 2015 年 「IRHOCS 2014 國際機器人實做競賽 機器手臂車組」 榮獲團體組 **冠軍**
- 2015 年 「2015 年智慧機器人創意競賽 工業機器人智慧應用創意組」 榮獲團體組 **冠軍**