ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

VISUAL SERVO-CONTROL APPLICATION IN A HUMANOID ROBOT USING DEPTH-CAMERA INFORMATION

M.Sc. THESIS

Arezou RAHIMI

Department of Electrical and Electronics Engineering

Control and Automation Engineering Programme

Thesis Advisor: Ass.Prof. Dr. Ali Fuat ERGENÇ
Thesis Co-advisor: Asst.Prof. Dr. Pınar BOYRAZ

JANUARY 2014

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

VISUAL SERVO-CONTROL APPLICATION IN A HUMANOID ROBOT USING DEPTH-CAMERA INFORMATION

M.Sc. THESIS

Arezou RAHIMI
(504111105)

Department of Electrical and Electronics Engineering

Control and Automation Engineering Programme

Thesis Advisor: Ass.Prof. Dr. Ali Fuat ERGENÇ
Thesis Co-advisor: Asst.Prof. Dr. Pınar BOYRAZ

JANUARY 2014

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

DERİNLIK KAMERA BİLGISİNİ KULLANARAK İNSANSI ROBOT'TA GÖRSEL SERVO-KONTROL UYGULAMASI

YÜKSEK LİSANS TEZİ

Arezou RAHIMI
(504111105)

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Kontrol ve Otomasyon Mühendisliği Programı

Tez Danışmanı: Yrd.Doç. Dr. Ali Fuat ERGENÇ
Tez Eş Danışmanı: Yrd.Doç. Dr. Pınar BOYRAZ

OCAK 2014

Arezou Rahimi, a **M.Sc.** student of ITU **Graduate School of Control and Automation engineering** student ID **504111105**, successfully defended the **thesis** entitled **"VISUAL SERVO–CONTROL APPLICATION IN HUMANOID ROBOT USING DEPTH-CAMERA INFORMATION"**, which she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Asst.Prof. Dr. Ali Fuat ERGENÇ** İstanbul Technical University
Co-advisor :	**Asst.Prof. Dr. Pınar BOYRAZ** İstanbul Technical University
Jury Members :	**Asst.Prof. Dr. Gülay Öke** İstanbul Technical University
	Asst.Prof. Dr. Utku BÜYÜKŞAHİN Yıldız Technical University
	Asst.Prof. Dr. Mustafa Fazıl SERINCAN Bilgi University

Date of Submission : 16 December 2014
Date of Defense : 20 January 2014

With my deepest gratitude to my family,
To my father, my life Hero that taught me how to stand up in difficulties.
To my mother, my angel that taught me to love unconditionally
To my brother, whom was always there for me

FOREWORD

I would never have been able to finish my dissertation without the guidance of my professors, help from friends, and support from my family.
I would like to express my deepest gratitude to my supervisors Dr. Ali Fuat Ergenc and Dr. Pinar Boyraz for their excellent guidance, caring, patience, and providing me with an excellent atmosphere for doing research. I do appreciate both but Meanwhile I must confess that special thanks goes to Dr. Boyraz for her patience and great help and showing me the way to grasp the knowledge in all of the subjects that I was facing difficulties.
I would like to thank Cihat Bora Yigit, who as a good friend was always willing to help and give his best suggestions on related issues and sharing his knowledge with me and helping me finishing my thesis. I have greatly benefitted from the guidelines offered by my friends in UMAY group. And my last and foremost appreciation is expressed toward all the dear ones whether far or near to whose generous support and continuous encouragement I owe.
I want to extend my sincere and warmest love and appreciate to my family for believing in me, for their continuous love and their supports in my decisions. Their love provided my inspiration and was my driving force. I owe them everything and wish I could show them just how much I love and appreciate them.

February 2014

Arezou RAHIMI
Control Engineer

TABLE OF CONTENTS

Page

ABBREVIATIONS

VS	**:** Visual Servo
IBVS	**:** Image Based Visual Servo
PBVS	**:** Position Based Visual Servo
ROS	**:** Robot Operation System
TOF	**:** Time Of Flight
RGB	**:** Red.Green.Blue
RGBD	**:** Red.Green.Blue.Depth
IR	**:** Infra Red
ASD	**:** Autistic Spectrum Disorders
DOF	**:** Degree Of Freedom

LIST OF TABLES

Page

LIST OF FIGURES

Page

VISUAL SERVO-CONTROL APPLICATION IN A HUMANOID ROBOT USING DEPTH-CAMERA INFORMATION

SUMMARY

Robotic systems have been increasingly employed in various industrial, urban, medical, military and exploratory applications during last decades. To enhance the robot control performance, vision data are integrated into the robot control systems. Using visual feedback has a great potential for increasing the flexibility of conventional robotic and mechatronic systems to deal with changing and less-structured environments. How to use visual information in control systems has always been a major research area in robotics and mechatronics. Visual servoing methods, which utilize direct feedback from image features to motion control, have been proposed to handle many stability and reliability issues in vision-based control systems.

Visual control is one of the key tools used by human beings to control their body motions to perform activities such as grabbing a cup of coffee and placing it on a table. Similarly, in the field of robotics it is often desirable to make use of visual data obtained from an imaging system to control the motion of a robotic manipulator in order to grab an object or to place the end effector tool in a certain position with respect to an object. To accomplish these tasks in a closed-loop manner, researchers have developed a number of visual servo control techniques that can provide a high degree of accuracy. In the following work an overview of the different visual servo control techniques reported in the literature is given, highlighting their main advantages and drawbacks. This thesis introduces Image-based Visual Servoing (IBVS) (to the contrary Position-based Visual Servoing (PBVS)) with eye- to- hand configuration that is able to reach an desired object.

Humanoid robots have a broad range of applications and a common attribute of all these applications is that the robot needs to operate in unstructured environments rather than structured industrial work cells. Motion control and trajectory planning for robots in unstructured environments face significant challenges due to uncertainties in environment modeling, sensing modalities, and robot actuation. This thesis attempts to solve a subset of these challenges.

This thesis focuses on visual servoing (VS) control systems, particularly on image-based visual servoing (IBVS) control structures. In IBVS, the error signal is computed in the image plane and the regulation commands are generated with respect to such error by means of a visual Jacobian.

The Image based visual servoing scheme is adapted for an eye-to-hand configuration and implemented with a 6 DOF Humanoid robot; depth camera (Kinect) instead of monocular or stereo vision methods is used to control the end effector of UMAY. Different formulations of the jacobian of the system are implemented and the performance of the system is tested through simulation environment.

DERİNLIK KAMERA BİLGISİNİ KULLANARAK İNSANSI ROBOT'TA GÖRSEL SERVO-KONTROL UYGULAMASI

ÖZET

İnsan hayatını kolaylaştırmak ve yeni imkânlar sunmak amacıyla günden güne artan bilgisayar teknolojisindeki araştırmaların büyük bir kısmı, insan müdahalesi olmadan kendi kendine hareket edebilen akıllı makineler geliştirmekle ilgilidir. İnsanoğlu, hayatını kolaylaştırmak amacıyla teknolojik alanda birçok atılımlar yapmıştır. Robot fikri de gerçekleştirilen ve hala güncelliğini koruyan bu atılımlardan biridir. Robotlar, endüstride, tıpta, haberleşmede ve daha birçok alanda kullanılmaktadır. Ayrıca, askeri uygulamalarda da robot kullanımı yaygındır. Robot teknolojisi, çağımız gelişim süreci içinde gelişen birçok bilimsel ve teknolojik olguların, robot adını verdiğimiz teknolojik ürünler üzerinde bütünleşmesi ve uygulamasını içerir.

İnsansı robotlar, şekil olarak insana benzemekle beraber, çoğu zaman insanların bulunduğu ortamlarda çalışmaları için tasarlanmış robotlardır. Bu durum insan-robot etkileşimini ve robotun daha önceden bilmediği ortamlarda çalışmasını gerektirir. Genelde endüstriyel robotlar fabrikaların önceden belirlenmiş ortamlarında ve tanımlanmış işler üzerinde çalışmaktadır. Buna karşın günlük hayatımızda kullanabileceğimiz robotlar için önceden tanımlanmış ortamlar yoktur. Robotların gündelik hayatımızda kullanımını sağlamak için robotların bulundukları ortamı görmeleri ve tanımaları gerekmektedir. Örnek vermek gerekirse, bir insanın daha önceden bilmediği bir masa üzerinde bulunan bir objenin, bu noktadan başka bir noktaya taşınması insanda bulunan görme yetisi sayesinde mümkün olmaktadır. Böyle bir uygulamanın robot üzerinde gerçeklenmesi de yapay bir görü sistemi kullanımı ile elde edilebilir. Bu durumda, endüstriyel robotlardan farklı olarak, robotun kolunu uzatacağı objenin cinsini ve yerini tahmin etmesi, bu koordinatlara doğru eklem açılarını kullanarak uzanması ve objeyi kavraması gerekir.

Yapay görme veya görüntü işleme teknolojisi, bir kameranın sensör olarak kullanımı ile dış dünya ile robot arasındaki bilgi akışının sağlanmasıdır. Bu noktada, bilginin görüntüden çıkarımı için belirli işlemlerin uygulanması gerekmektedir. Bu işlemler görüntü işleme algoritmaları olarak adlandırılır. Görüntü işleme algoritmalarının robot gibi akıllı makineler ile birlikte kullanılması, insan yaşamını daha da kolaylaştırdığı için günümüz teknolojik gelişmeleri arasında önemli bir yer tutmuştur. Bir sensör olarak kamera, diğer algılayıcı teknolojilerinden farklı olarak ucuzdur, hafiftir, ve kapladığı alana oranla yüksek miktarda bilgi verir. Benzer şekilde insan üzerinde bulunan 5 duyu organından biri de gözdür ve temel olarak robotlarda, yapay görü ve görüntü işleme uygulamaları insan gözünü taklit etmeye yöneliktir. Renk algılama, nesne tanıma, hareket algılama, şekil algılama gibi uygulamaların dışında, bir nesnenin derinliğinin algılanması da görüntü işleme algoritmaları tarafından gerçeklenebilir. Bugüne kadar derinlik algılayabilmek için en az iki kameralı sistemler kullanılırken, gelişen teknoloji ile derinlik algılayıcı sensör ve kameranın entegre çalıştığı sistemler gerçekleştirilmiştir.

Bu tez kapsamında, UMAY (uyarlamalı mekanik ara yüz) projesi için tasarlanan 6 serbestlik dereceli kol ve gövde benzetim ortamında kullanılmıştır. UMAY projesi otistik çocukların eğitim ve rehabilitasyon süreçlerinde kullanılmak üzere geliştirilen bir insansı robot projesidir. UMAY projesi ile ilgili çalışmalar İstanbul Teknik Üniversitesi bünyesinde faaliyet gösteren Mekatronik Eğitim ve Araştırma Merkezi Laboratuvarları'nda yapılmakta olup, proje ile ilgili çalışmalar halen devam etmektedir.

Bu tez kapsamında, bir insansı robot projesi olan UMAY'ın 6 serbestlik dereceli kolu ve Kinect derinlik algılayıcı sensör ve kamera sistemi kullanılarak görsel servo kontrolü benzetim ortamında yapılmıştır.

Görse servo kontrol yardımı ile robot kolu kontrolünde literatürde pek çok uygulama mevcuttur. Bunlar; sistem hata sinyali (system error signal based), kamera robot entegre düzeneği (camera-robot setup based) ve kamera sayısı tabanlı (number of cameras based) çalışma başlıkları altında incelenebilir.

Sistem hata sinyali tabanlı çalışmalar iki başlık altında toplanabilir: Konum bazlı ve görüntü bazlı. Konum bazlı uygulamalar, görüntülerden özniteliklerin çıkarımı (feature extraction), hedef nesnenin konum ve oryantasyonun kestirimi, konum ve oryantasyondaki kestirim hatasının geri besleme yardımıyla azaltılmasını sağlayacak hesaplamalar olarak gösterilebilir. Görüntü bazlı uygulamalar ise, görüntülerden özniteliklerin çıkarılması ve bu öznitelikler üzerinden kontrol değerlerinin hesaplanması olarak karşımıza çıkar. Bu uygulamalar iki boyutlu olduğu için kalibrasyon hataları daha az karşılaşılır.

Kamera robot entegre düzeneğinde iki yaklaşım bulunmaktadır. Bunlar, kameranın robot koluna monte edildiği (eye-in-hand) ve kameranın robot kolu ve hedef nesneleri beraber görebileceği (eye-to-hand) şekilde kameranın konumlandırılmasını sınıflandırmaktadır. Bu kontrol yapılarında açık (EOL) ve kapalı (ECL) kontrol teknikleri uygulanmaktadır. Kamera kol üzerine monte edildiğinde sadece nesne görülebildiği için açık çevrim kontrol kullanılabilir. Buna karşın kamera kolu ve hedef nesneyi kontrol işlemi süresince görebilecek bir konuma yerleştirildiği takdirde, kapalı çevrim kontrol uygulanabilir olur.

Kamera sayısı tabanlı çalışmalar, tek kamera (monocular), çift kamera (binocular) veya ikiden fazla kameranın kullanıldığı yaklaşımlar şeklinde sıralanabilir. Tek kameralı sistemler, görsel bilgiyi çıkarımı için gerekli işlem zamanını en aza indirir. Nitekim nesne modeli bilinmediği için derinlik bilgisinin kaybolması görsel servo ile kontrol işlemini kısıtlar ve kontrol sistemi tasarımını karmaşık hale getirir. Çift kameralı sistemler ise, nesne ve sahne ile ilgili 3 boyutlu bilgiyi sağlar. Bu metot iki farklı görüntüleme cihazı kullandığı için stereo görüntü sistemleri tek kameralı sistemlere göre iki kat daha fazla işlem yüküne sahiptir.

Bu çalışmada, görüntü temelli servo kontrolü kapalı bir kontrol çevrimi içerisinde kullanılarak, benzetim ortamında 6 serbestlik dereceli bir kolun, daha önceden tanımlanmamış bir masa üzerinde bulunan bir objeye ulaşması sağlanmıştır.

Masa üzerinde bulunan nesneye erişmek için tahmin edileceği üzere hedef nesnenin mutlak koordinatlarda derinliğinin saptanması çok büyük önem arz etmektedir. Görsel etkileşim matrisi olarak isimlendirilen matrisin hesaplanmasında derinlik bilgileri kullanılır. Literatürde bulunan çalışmaların aksine, bu çalışmada stereo ve tek kamera kullanımının dezavantajları bulunması nedeniyle, yeni bir yöntem ile

Kinect sistemi kullanılmıştır. Bunun sonucunda, derinlik hesaplamasında işlem zamanı literatürde verilen çalışmalara oranla bir düşüş göstermiştir.

Kapalı çevrim kontrol uygulanabilmesi için Kinect robot gövdesi üzerine yerleştirilmiştir. Bu sayede hem hedef nesne, hem de robot kolu işlem boyunca Kinect tarafından görülebilir, bu durum kameranın el üzerine konulduğu durumların aksine işlem süresi içerisinde hedef nesnenin görüntüden çıkması problemini engeller. Alışılmışın aksine, görsel etkileşim matrisi bu uygulamada kapalı çevrim kontrol içerisinde kullanılmaktadır.

Benzetim için Ubuntu işletim sistem ile çalışan i5 işlemcili 4 GB belleği olan bir bilgisayar kullanılmıştır. Solidworks ortamında üç boyutlu tasarımı yapılan robot ve kol modeli, URDF (universal robot description format) formatına çevrilerek benzetim ortamına aktarılmıştır. Kontrol algoritmaları ve robot modelinin dinamik simülasyonu robot işletim sistemi ROS (Robot Operating System) groovy versiyonu ve gazebo programı (1.5 versiyonu) kullanılarak yapılmıştır. Görüntü işleme için OpenCv kütüphanesi kullanılmıştır. Kontrol uygulaması python 2.7 ve C++ programlama dillerinde yazılmıştır.

Kinect sistemi yardımıyla benzetim ortamında derinlik algılanması ve nesnelerin tanımlanması sağlanmıştır. Literatürde kullanılan derinlik algılama yöntemleri arasında reflektif yöntemler içerisinde bulunan optik aktif ve pasif yöntemler bulunmaktadır. Stereo görüntü işleme pasif bir yöntem iken, Kinect, aktif bir şekilde ışık kodlaması ile üçgenleştirme metodunu kullanarak derinlik tespiti yapar. Bu sensör, içerisinde bir adet kızılötesi kamera, bir RGB (kırmızı yeşil mavi) kamera ve bir adet de kızılötesi projektör bulundurur.

Bu çalışmada; sabit derinlik, değişken derinlik ve bunların kombinasyonları kullanılarak, yukarıda ifade edilen benzetim şartlarında robotun kolun uç eyleyicisinin masa üzerine konulan hedef nesneye ulaşması başarıyla sağlanmıştır. Burada sabit derinlik olarak uç eyleyicinin başlangıç pozisyonunda hedef nesneye olan uzaklığın tüm benzetim boyunca sabit olarak kullanılması şeklindedir. Bu nedenle tüm benzetim boyunca uç eyleyicinin üç boyutlu uzaydaki hızı sabit kalmaktadır. Diğer yöntemlerde her iterasyonda uç eyleyici ile hedef nesne arasındaki derinlik bilgileri tekrar algılanmış ve bu bilgiler ışığında kontrol gerçekleştirilmiştir. Burada uç eyleyici için değişken hızlar elde edildiği görülmüştür. Burada kullanılan iki yöntem MPJ (mean of pseudo-inverse Jacobian) ve PMJ (pseudo-inverse of mean Jacobian) olarak adlandırılır ve benzetimler göstermiştir ki MPJ kullanımı bu uygulama için daha iyi sonuçlar vermektedir.

1. INTRODUCTION

Robotics is a branch of technology with designs, construction, and application of robot along with the computer system for its functions. It can take place of humans and function automatically. Moreover, it may be required to work like humans. Today the researchers of robotics are dealing with a new challenge that is humanoid robotics. A long-standing desire that human-like robots could coexist with human beings has made the researchers think that the humanoid robotics industry will be a leading industry in the twentieth- first century This thought comes from the fact that technology is finally getting ready for this purpose. Fastest micro-processors, super computers, high-torque servo-actuators, precise sensors along with new advances in control techniques, artificial intelligence and artificial sound/vision recognition, all embedded in better mechanical designs made the researchers and entrepreneurs believe that this dream might become true in a near future. The study of humanoid in robotics field is related to understand the interaction between the robot, human and the environment. The humanoid robotics inspires communal connection like motion or any supportive task similar to physical dynamics.

The use of robotic systems has remarkably contributed to increase the speed and precision of automated tasks. However, generally such robot systems require a detailed description of the workspace and manipulated objects. This is not an issue when the required task employs only fully characterized objects, within a completely known environment as is the case with most industrial assembly lines. However, it has been widely discussed that there exists an inherent lack of sensory capabilities in modern robotic systems which make them unable to cope with new challenges such as an unknown or changing workspace, undefined locations, calibration errors and so on.

In response to this challenge, visual servoing (VS) was born. It is said that the versatility of a given robot system can be greatly improved by using artificial vision techniques. VS emerges naturally from our own human experience and from observing other living beings which are able to execute complicated tasks thanks to

their visual systems although they might be sometimes primitive. Through our sense of vision, humans are able to measure the environment and gather relevant data which together with other reasoning process, contributes to the ability of performing complicated tasks. So far, when it is required that a given robot executes different sort of tasks, considerable human and economic efforts should be spent in robot reprogramming, relocation, redesign of the workspace and re-characterizing of the objects and so on, which evidently increases the labor and production costs.

A "*Visual Servoing System*" (VS) is a feedback control system based on visual information. The VS is essential for autonomous robots working in unknown or unstructured environments. In general, this system is composed of one or more cameras, a processing or computing unit, and specific image processing algorithms to control the position of the robot's end-effector relative to the object or work piece as required by the task. Visual servoing systems have been increasingly used in control of robot manipulators that is based on visual perception of robot and object location. It is a multi- disciplinary research area spanning computer vision, robotics, kinematics, control and real-time systems.

It is expected that equipping the robot with higher abilities to interact directly with the environment such as visual analysis capabilities, more complex tasks would be effectively performed by the robot, reducing the time spent for redesign times eventually saving money.

1.1 Purpose Of Thesis

Humanoid robots have a broad range of applications and a common attribute of all these applications is that the robot needs to operate in unstructured environments rather than structured industrial work cells. Motion control and trajectory planning for robots in unstructured environments face significant challenges due to uncertainties in environment modeling, sensing modalities, and robot actuation. This thesis attempts to solve a subset of these challenges.

There is a strong demand to use vision-based robots in everyday environments, because vision adds versatility to a robot. Real-time motion control of robots from visual feedback, visual servoing, is distinct from regular robot control in that it uses the (projective) camera coordinates instead of a fixed Euclidean robot base frame.

Visual servoing is a well-studied framework for real-time vision-based motion control of robots [17, 18, 19]. Many elementary robotic tasks, such as manipulation, benefit from visual servoing [20]. A formal discussion of the visual servoing problem, along with a comprehensive review of the literature, the available approaches, their strengths and limitations will be presented in next chapters. Here, we briefly present where this thesis stands within the broad visual servoing literature

In this work a new implementation approach for visual-servoing in ***six*** degrees of freedom (DOF) is described. In general approach of visual servo controls, the procedures of image-based visual-servoing (IBVS) need depth information, which plays a crucial role in the overall algorithm performance. The depth information has to be obtained fast in each iteration for calculating the interaction matrix .The motivation of this paper is not similar to existing work describing visual servoing method, which uses eye-in-hand method, using IBVS scheme which is adapted for 'eye-to-hand' configuration. The implementation is achieved with an optical depth sensor to control the position and orientation of its end effector. Obtaining the depth information directly from Kinect accelerates the solution, and reduces the computational cost of the control algorithm. The framework is implemented on the UMAY humanoid robot. The implementation on UMAY is performed using the ROS as software architecture. The results presented show simulations on Gazebo. The robot is tested with different rehabilitation-play scenarios with the applied method.

1.2 Literature Review

The main application of visual servoing in industrial robotics concerns with the control of the end-effector pose (position and orientation) with respect to the pose of objects or obstacles, which can be static or dynamically moving in the workspace of the robot. These robotic systems can perform several exemplary tasks such as positioning or moving some objects, assembling and disassembling mechanical parts, paintings, welding in a workspace, which may contain static and\or dynamical obstacles or targets [1].

In robot visual servoing system, the control of the pose is determined using synthetic "*Image Features*" extracted from a sequence of images captured with imaging devices [2], [3]. These image features are provided by an imaging device e.g. one or more cameras, mounted on the end effector of the robot or in a fixed position with

respect to the robot workspace. you can see industrial robot using visual feedback in figure 1.1. [4], [5].

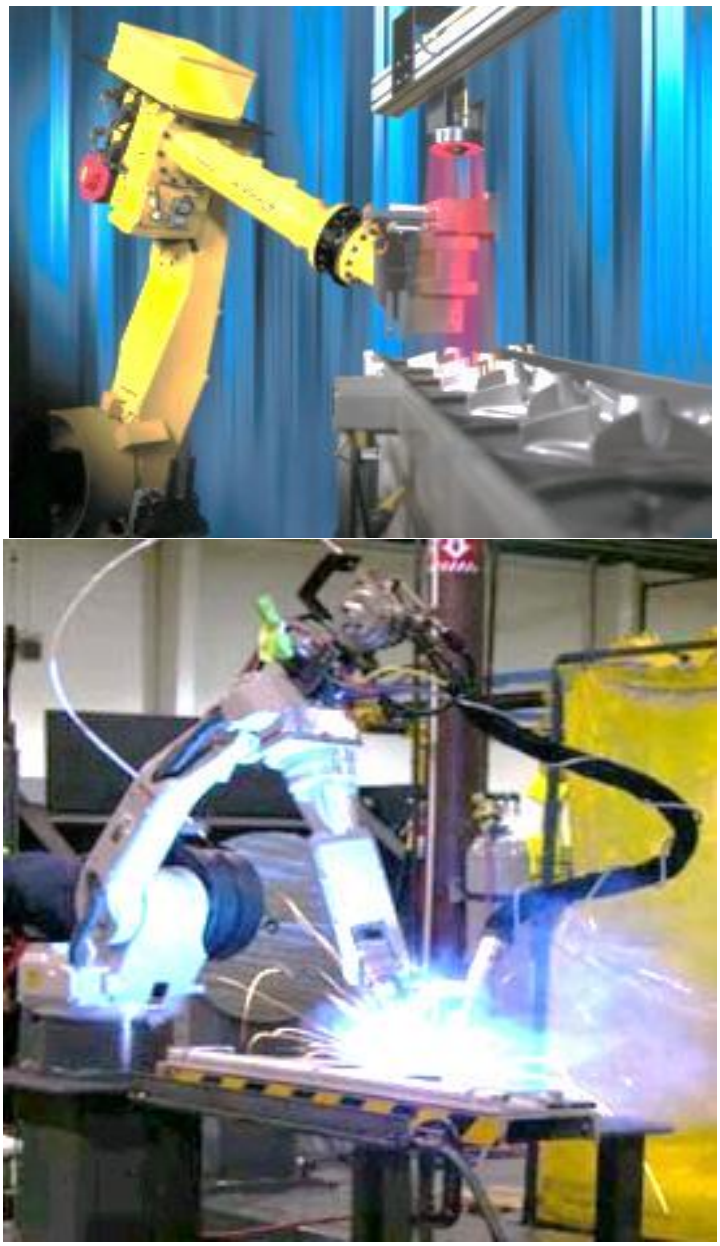


Figure 1.1 : Robots using visual feedbacks to perform various tasks.

Visual Servoing tends to be widely used in medical and surgical applications to position instruments or perform the medical operations. For instance "*Laparoscopic Surgery"* is minimally invasive, which means it only needs several small incisions in the abdominal wall to introduce instruments such as scalpels, scissors, etc., and a laparoscopic camera so that the surgeon can operate by just looking at the camera images. To avoid the need for another assistant and to free the surgeon from the control task, an independent system that automatically navigates the laparoscopy

equipment is highly desirable. Several researchers have tried to use visual servoing techniques to guide the instrument during the operation .see Figure 1.2 [6], [7].

Figure 1.2 : Application of Visual Servoing in Medical Robotic Systems.

Control and guidance of unmanned vehicle systems is another example of using visual servoing technique for the exploration or reconnaissance operations. The pose of an unmanned ground vehicle (UGV) is typically required for autonomous navigation and control [8]. Often the pose of an UGV is determined by a global positioning system (GPS) or an inertial measurement unit (IMU). However, both GPS and IMU have many limitations such as signal availability and in many cases high costs. Given recent advances in image processing technology, an interesting approach to overcome the pose measurement problem is to use a visual servoing system, Figure 1.3 [9].

Figure1.3 : Using vision system in a mobile unmanned vehicle.

In another application of visual servoing systems, space robots are used to perform autonomous on-orbit servicing, which includes approaching and docking to a target satellite and grasping some complex parts for the purpose of refueling and servicing see Figure 1.4 [10].

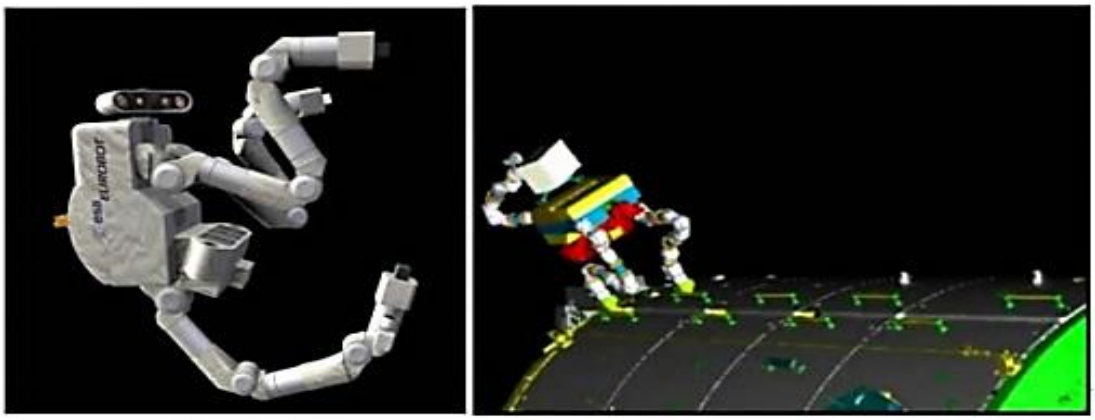


Figure 1.4 : Space robotic system in a mission of on-orbit servicing.

One of the most challenging technological endeavors of human kind has been giving the capabilities of gathering complex information on the environment to machines in order to interact with the environment in an autonomous manner. The two most important senses which provide sufficient environmental information for human to perform interaction tasks are the tactile and the visual senses. The devices that are able to partially imitate these human senses are the force sensors and the visual sensors, respectively. The visual sense is often lacking in many human-made machines. In fact, without visual information, manipulating devices can operate only in "structured" environments, where every object and its relative position and orientation is known *a priori*. With the increase of real-time capabilities of visual systems, vision is beginning to be utilized in the automatic control as a powerful and versatile sensor to measure the geometric characteristics of the work piece as well as its dynamic position which is an uncertain information representing 'unstructured environment'.

The goal of many robotic applications is to place the robot at a desired configuration to manipulate an object in an environment. Computer vision adds versatile sensing to robotic manipulation since it can sense the position of the object as well as tracking it dynamically. Although the promising aspect of visual information, the early approaches to vision-based robotics included only monitoring and inspection applications, where visual feedback is not used in a closed- loop control scheme.

To place the end-effector of the manipulator at a desired position with respect to the object, the rigid-body transformation between the object and the end-effector must be known.

Figure 1.10 shows a manipulator with a camera and an object and the corresponding coordinate frames [25].

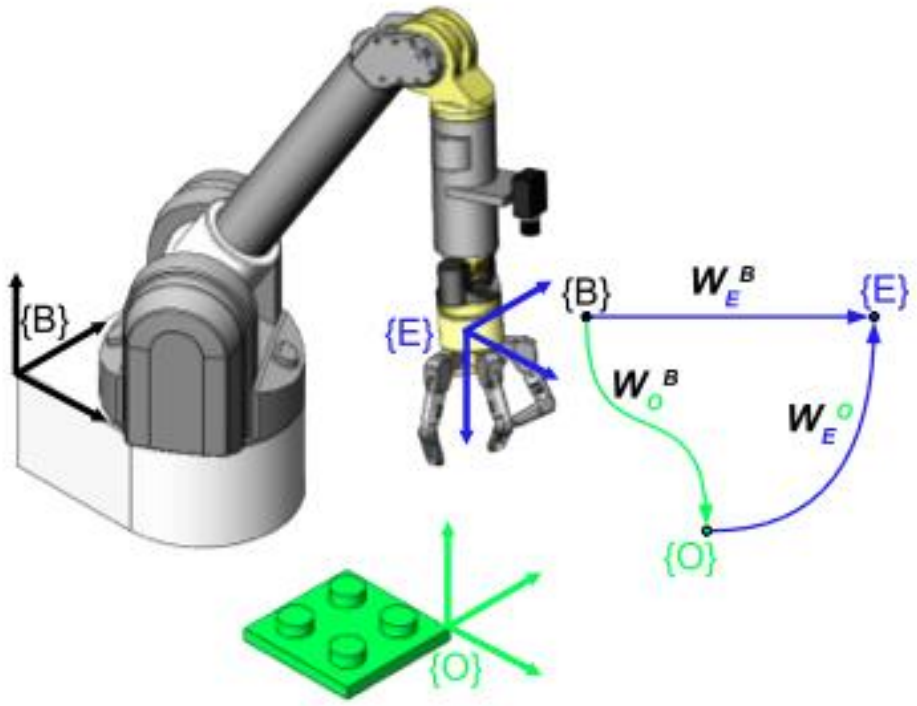


Figure1.5 : Open-loop robot control without using feedback from the camera.There are three frames:robot base, object, and end-effectore.

Let the robot base frame be denoted by {B}, the frame at the end-effector by {E}, the object frame by {O}, the transformation from {B} to {O} by w_o^B, the transformation from {B} to {E} by w_E^B, and the transformation from {O} to {E} by w_E^O. Given w_o^B (e.g., object on a fixture with calibrated distance from the base) and the desired w_o^B, one can calculate w_E^B and then by solving the inverse kinematics problem, find the robot configuration.Despite the limiting assumption to calculate w_o^B and perfect robot calibration *apriority*, many industrial applications such as factory automation and visual part inspection still use this open-loop framework. It is clear that this approach is limited to very structured environments and does not apply to unstructured settings.The forward kinematics transformation is denoted by w_E^B, the base to object transformation by w_o^B, and object to end-effector by w_E^O. Robot configuration can be

updated by solving the inverse kinematics from known w_o^B (object on a previously known fixture in structured settings) and w_E^O (user-defined).

The history of visual servoing goes back to the seventies. In the early 1970s, Shirai and Inoue [21] described how *visual feedback*, as the use of vision in the feedback loop, can increase the accuracy in tasks. The term "*visual servoing*" was first introduced by Hill and Park [22] in 1979. Prior to the introduction of this term, the less specific term *visual feedback* was generally used. Afterwards, considerable researches [23, 24] have been performed on the development of visual servoing control systems. The analytical complexity of robot control systems and also processing vision data have made the vision-based control problem challenging. Recently, both computers and video cameras are fast and advanced and consequently are increasingly used as robotic sensors in feedback control systems. Therefore, the control of robots employing visual feedback is now more practically feasible.

To add flexibility to vision-based robots, visual feedback can be used making the system close-loop controlled. Figure 1.11 shows the addition of a camera frame {C} to the previous vision-based manipulator shown in Figure 1.10 [25].

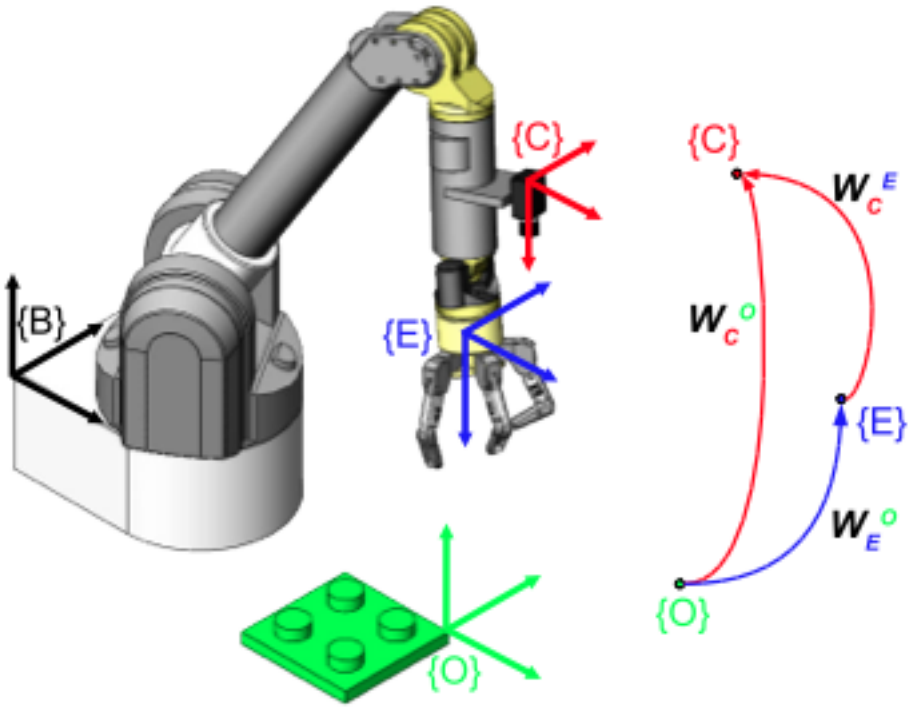


Figure1.6 : Closed-loop robot control using the relative object-to-camera pose.

The other transformations of interest are the object-to-camera transformation w_C^O

and the end- effector-to-camera transformation w_C^E. Addition of a camera sensor enables bypassing of the robot base frame to calculate the relative object to robot transformation. In particular, there is no need to place the object on a known fixture if the visual feedback is employed.

With a feedback signal from the camera, the robot base frame and fixed object fixture can be bypassed. The other three frames are the object, the end-effector, and the camera. Transformation w_C^E takes the end-effector frame to the camera frame and is found by calibration. Transformation w_C^O denotes the relative object-to-camera pose.

One strategy is to compute transformation w_E^O from a calibrated transformation w_C^E and estimate the relative camera-to-object relative pose w_C^O from pose estimation algorithms. Once the transformation w_E^O is computed, the robot can be moved towards the desired pose without further pose estimation. This is called the static "look and move" control architecture [26].

In general, visual feedback is provided by one or more cameras that are either rigidly attached to the robot (eye-in-hand configuration as in Figures 1.11 or static in the environment looking at the robot motions (eye-to-hand configuration, not shown in figures). The initial and desired states define an error, which is to be minimized and regulated to zero at the desired state.

Computer vision algorithms are used for the tracking of visual features on the object. The visual features can be used to compute the relative object-to-camera pose or to compute an error in the image space. The typical visual features for tracking are either geometric primitives or appearance-based. Examples of geometric features are dots, lines, contours, and their higher order moments [27]. An example of an appearance-based feature is known as the Sum of Squared Distances [28].

Visual servoing is a framework where real-time visual feedback is used to control a robot to a desired configuration [21, 22]. Visual servoing is also studied as vision-based motion control and robotic hand-eye coordination with a feedback.

Depending on the type of the error in the control law, one can classify the visual servoing system to three main classes: position-based visual servoing (PBVS) or 3D-visual servoing [29], image-based visual servoing (IBVS) or 2D visual servoing [30], and hybrid visual servoing [31, 32]. Stability analysis and performance studies of

these approaches are available in the literature [33, 34, 35].

1.3 Humanoid Robot

Humanoid robots are meant to communicate and interact with humans are different from industrial robots in terms of their set of requirements. In humanoid design, the primary concern is to make sure that no user of this type of robot will come to harm. The robot needs “a motion space that corresponds to that of human beings and a lightweight design.” The robot must be somewhat humanlike in appearance and dexterity. “...its kinematics should be familiar to the user, its motions predictable, so as to encourage inexperienced persons to interact with the machine.” [11] like KASPAR as you see in Figure 1.7 , 1.8 [12].

Figure 1.7 : KASPAR, An Expressive, Interactivve Robot.

Figure 1.8 : Health Care Robot.

To enhance the robot control performance, vision data are integrated into the robot control systems [13]. Early works on using visual servoing for enabling a robot to grasp objects are reported in [14] and [15]. Since then, it has been utilized in much more complicated scenarios. The realization of robots similar to humans has been the goal of many research groups around the world. The replication of the human visual sense is one of the most important aspects of such goal. Also with the increase of real time capabilities of visual systems, vision is beginning to be utilized in the automatic

control as a powerful and versatile sensor to measure the geometric characteristics of the work piece.

1.3.1 Humanoid robot UMAY

Humanoid robots are designed with a human form factor, which defines similarity between human and robot behaviors. The desire for this kind of design is rooted in the need to bring robots into everyday life enabling them to work in the environments in which people work and live naturally. Many of the earliest motivations for developing humanoids, centered on creating robots that can play a role in the daily lives of people. Today, humanoid robots are being developed to provide the elderly with assistance in their homes and to support medical care. Humanoid UMAY aims to help incremental rehabilitation of children with Autistic Spectrum Disorders (ASD). Autism is a disorder that primarily affects the development of social and communication skills. Interacting with humanoid robots that provide interaction with these children has been shown to improve the communication skills of autistic children [81]. Humanoid UMAY is designed to be used in clinical therapies for the children with ASD using some rehabilitation tools or toys. In particular, early intervention and continuous care provide significantly better outcomes. Currently, there are no robots capable of meeting these requirements that are both low-cost and available to families of autistic children for in- home use. Humanoid robot UMAY is designed to obtain a low-cost and accessible platform for the in-home rehabilitation of children with ASD. Furthermore, the visual sense is often lacking in many existing human-made machines. In fact, without visual information, manipulating devices can operate only in “structured” environments, where every object and its relative position and orientation is known a priori. In UMAY, computer-vision interface is considered as a core capability. Human-robot interaction and learning module in UMAY need to use visual servo control as a sub-module to interact with its environment in a more controlled way.

1.3.2 System overview

UMAY is a humanoid platform that has 6 DOF arms with all revolute joints, a special hybrid neck mechanism [16], and a moving base platform. The visual servoing framework has been implemented for UMAY Robot that is shown in Figure 1.9.

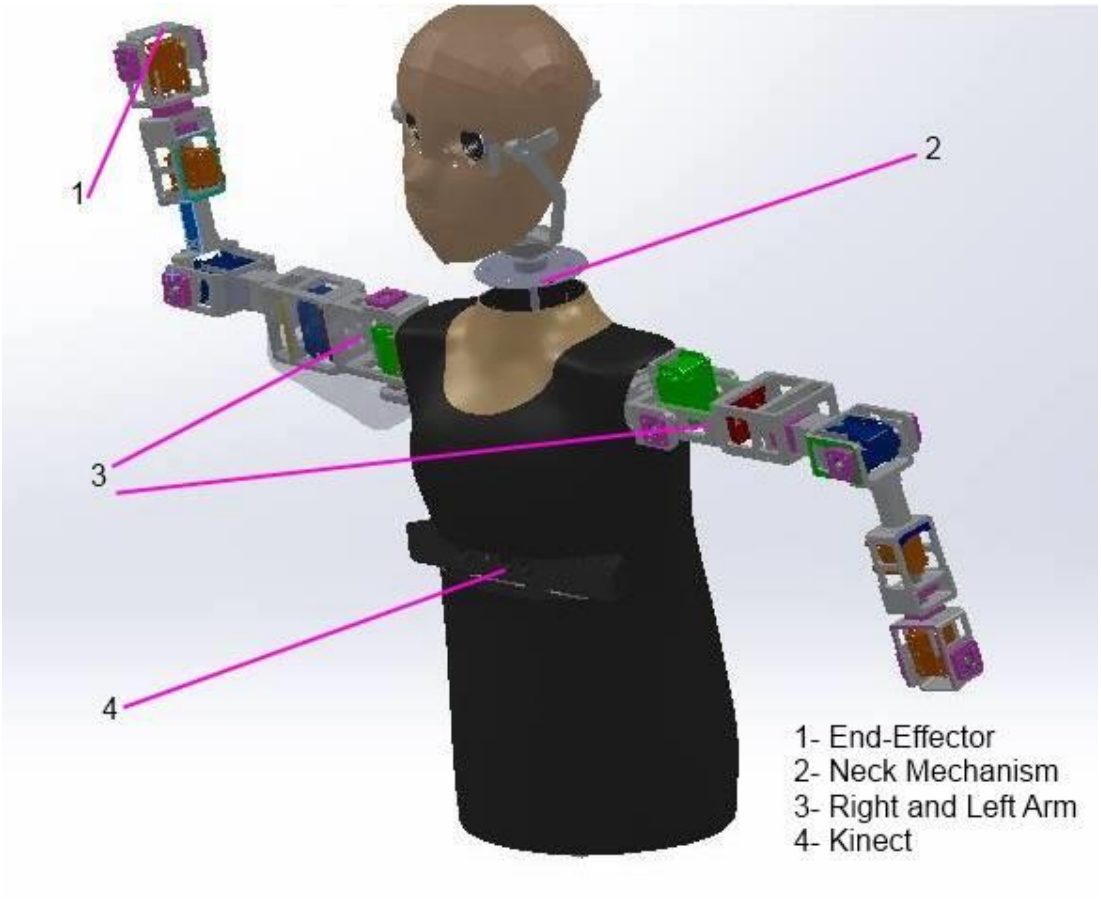


Figure1.9 : An overview of UMAY robot with sensor position indicated.

The Denavit-Hartenberg (D-H) table is developed to solve the forward and inverse kinematics problems.

The parameters used in D-H table are described as below.

a_i :Off set distance between two adjacent joint axes.

d_i :Translation distance between two incident normal of a joint axis.

α_i : Twist angle between two adjacent joint axes.

θ_i: Joint angle between two incident normal of a joint axis.

In order to describe the kinematics and dynamics of robot Umay we use The Denavit-Hartenberg parameters as seen in Table 1. 1.

Table 1.1 : D-H parameters of Umay.

i	Alpha (i-1)	a(i-1)	d(i)	teta(i)
1	0	0	0	T1
2	-90	0	20	T2
3	90	5	264	T3
4	-90	0	0	T4
5	-90	0	212	T5
6	90	0	0	T6

1.3.3 Vision system

The vision system is a very important and essential device helping the robot localize and recognize the object. The vision system used in this dissertation helps calculating the position of the target to grasp and also the relative position of the robot to the target. Moreover, it will serve as the input signals for the visual-servo controller in this work. In this thesis, unlike other work, a new implementation including the depth camera was used. On contrary to common eye in hand method that vision system attached to the end-effector, a depth camera is located in the middle of torso of the robot, using for recognizing both objects and the end effector calculating the relative position of the object to the end effector of the robot arm as shown in Figure 1.8.

1.3.4 Simulation environment

A computer with an Intel i5 processor and 4 GB RAM was used for the simulations described below.

The operating system of the computer was Ubuntu 12.04 and runs Robot Operating System (ROS) (Groovy version) and Gazebo (1.5 version) as the dynamic simulator.For the programming environment several languages are used such as C++ and Python and OpenCV (2.6.1) computer vision library was employed. Gazebo publishes the depth and RGB images which are taken from the simulated depth camera. In addition to this, Gazebo publishes the 'joint states' *topic* which includes kinematic states of the joints. Since the robot arm has 6 degree of freedom, there are 6 joint velocity controllers which can be seen from the left part of the figure. 'Image

processing' node, subscribes the RGB images topic and finds the red object and green end effector. It publishes the coordinates in the image frame. 'Visual_servo_controller' node implements the Image based visual servo controller (IBVS) which is explained in this thesis 'Motor_commands_sender' subscribes the joints states in order to calculate Jacobian matrix and visual servo controller for taking the desired velocities of the end effector. It calculates the velocities of all motors. Joint controllers control the motors speeds. The depth camera runs on 1 Hz and other nodes run on 100 Hz. For synchronization of depth and RGB images, the 'message filters' library was used.

The figure 1.9 shows the connection between the ROS nodes.

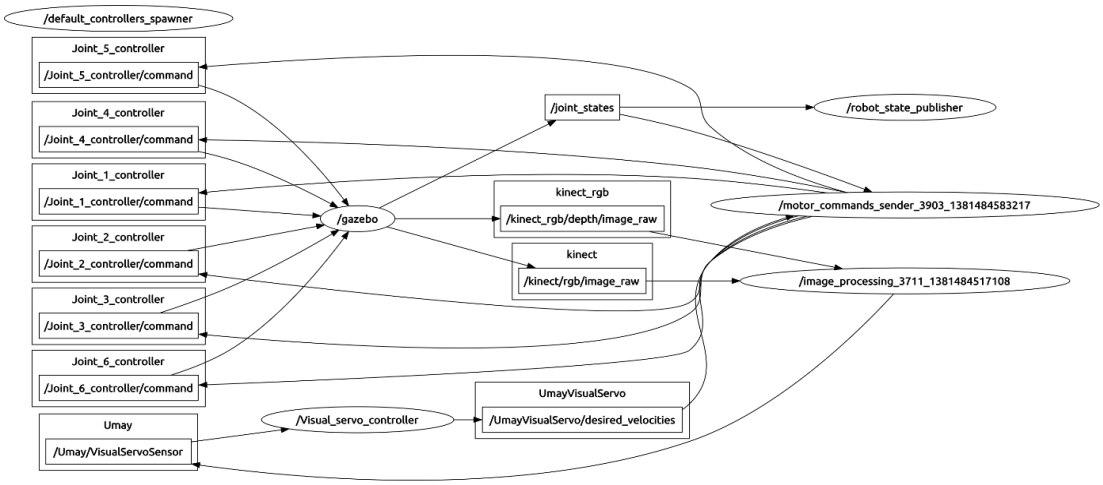


Figure 1.10 : ROS nodes conections.

1.3.4 Problem statement

Humanoid robots have a broad range of applications and a common attribute of all these applications is that the robot needs to operate in unstructured environments rather than structured industrial work cells. Motion control and trajectory planning for robots in unstructured environments face significant challenges due to uncertainties in environment modeling, sensing modalities, and robot actuation. This thesis attempts to solve a subset of these challenges.

There is a strong demand to use vision-based robots in everyday environments, because vision adds versatility to a robot. Real-time motion control of robots from visual feedback, visual servoing, is distinct from regular robot control in that it uses the (projective) camera coordinates instead of a fixed Euclidean robot base frame.

Visual servoing is a well-studied framework for real-time vision-based motion control of robots [17, 18, 19]. Many elementary robotic tasks, such as manipulation, benefit from visual servoing [20]. A formal discussion of the visual servoing problem, along with a comprehensive review of the literature, the available approaches, their strengths and limitations will be presented in next chapters. Here, we briefly present where this thesis stands within the broad visual servoing literature

In this work a new implementation approach for visual-servoing in ***six*** degrees of freedom (DOF) is described. In general approach of visual servo controls, the procedures of image-based visual-servoing (IBVS) need depth information, which plays a crucial role in the overall algorithm performance. The depth information has to be obtained fast in each iteration for calculating the interaction matrix .The motivation of this paper is not similar to existing work describing visual servoing method, which uses eye-in-hand method, using IBVS scheme which is adapted for 'eye-to-hand' configuration. The implementation is achieved with an optical depth sensor to control the position and orientation of its end effector. Obtaining the depth information directly from Kinect accelerates the solution, and reduces the computational cost of the control algorithm. The framework is implemented on the UMAY humanoid robot. The implementation on UMAY is performed using the ROS as software architecture. The results presented show simulations on Gazebo. The robot is tested with different rehabilitation-play scenarios with the applied method.

2. GENERAL CONCEPT IN VISUAL SERVOING OF ROBOT ARM

This chapter develops the basic idea of controlling a robot using the image provided by a camera. Although many authors argue that this concept has intuitively emerged directly from our human nature, it is obvious that not only humans but many living beings acknowledge their environment through a great deal of visual information. The advantages are evident because visual sensing facilitates adaptation and other intelligent behavior, which eventually have evolved resulting in well-developed systems to which humans can attribute their success.

The chapter starts by considering a 6-DOF robotic arm, some basic information about Robot is given then a short and concise summary of visual servoing concepts is presented. Also in this chapter, some guidelines about camera and simulation environment are mentioned.

2.1 Basics About Robots

2.1.1 Links and frames

The individual bodies that together form a robot are called *links*, and they are connected by *joints* (also called *axes*). Generally a robot with n degrees of freedom has n + 1 links. The base of the robot is defined as link 0, and the links are numbered from 0 to n. The robot has n joints, and the convention is that joint i connects link i−1 to link i. The robot can be seen as a set of rigid links connected by joints, under the assumption that each joint has a single degree of freedom. The total degrees of freedom for the robot are however equal to the degrees of freedom associated with the moving links minus the number of constraints imposed by the joints. Most of the industrial robots have six degrees of freedom, which makes it possible to control both the position and orientation of the robot tool.

The dynamics of the robot is coupled, which means that a joint provides physical constraints on the relative motion between two adjacent links. When relating the

links and their motions to each other, *coordinate frames* (coordinate systems) are used. Frame 0 is the coordinate frame for the base of the robot (joint/link 0), and frame i is placed at the end of link i, which means in joint i.

2.1.2 Pairs and joints

The kind of relative motion between links connected by a joint is determined by the contact surfaces, called *pair elements.*

Two pair elements form a *kinematic pair*. If the two links are in contact with each other by a substantial surface area, it is called a *lower pair*. Otherwise, if the links are in contact along a line or at a point, it is called a *higher pair*. A revolute joint, prismatic joint, cylindrical joint, helical joint, spherical joint and plane pair is all lower pairs. The frequently used universal joint is a combination of two intersecting revolute joints. Examples of higher pairs are gear pair.

All types of joints can be described by means of revolute or prismatic joints, both having one degree of freedom. Prismatic joint can be described by a cube with side d, resulting in a translational motion. As Humanoid robot UMAY has revolute joint, in this thesis it is sufficient to know that a revolute joint has a cylindrical shape, where the possible motion is a rotation by an angle φ. Further work is therefore only applied to revolute or prismatic joints. [82]

The joint variable q is the angle φ for a revolute joint, and the link extension d for a prismatic joint. The joint variables $q_1, \ldots, q_n$ form a set of generalized coordinates for an n-link serial robot, and are used when choosing general coordinate frames according to the convention by Denavit and Hartenberg (1955).

2.1.3 Kinematics

Kinematics describes the movements of bodies without considerations of the cause. The relations are fundamental for all types of robot control and when computing robot trajectories (Bolmsjö, 1992). More advanced robot control involves for example moments of inertias and their effects on the acceleration of the single robot joints and the movement of the tool. Dynamic models are then required., which are briefly described in next Sections.

In kinematic models, position, velocity, acceleration and higher derivatives of the position variables of the robot tool are studied. Robot kinematics especially studies

how various links move with respect to each other and with time. This implies that the kinematic description is a geometric one. (Corke, 1996a)

Using coordinate frames attached to each joint, the position p and orientation φ of the robot tool can be defined in the Cartesian coordinates x, y, z with respect to the base frame 0 of the robot by successive coordinate transformations. This results in the relation

$$p_0 = R_0^n p_n + d_o^n \qquad (2.1)$$

Where p_0 and p_n are the position of the tool frame expressed in frame 0 and tool frame n, respectively. The rotation matrix R_0^n describes the rotation of the frame n with respect to the base frame 0, and gives the orientation φ. The vector d_o^n describes the translation of the origin of frame n relative to the origin of frame 0.

The rigid motion can be expressed using homogeneous transformations H as in

$$p_0 = \begin{pmatrix} p_0 \\ 1 \end{pmatrix} \qquad p_n = \begin{pmatrix} p_n \\ 1 \end{pmatrix} \qquad (2.2)$$

$$p_0 = H_0^n p_n = \begin{pmatrix} R_0^n & d_0^n \\ 0 & 1 \end{pmatrix} p_n \qquad (2.3)$$

It must be mentioned that there is a variety of formulations of the kinematics, based on vectors, homogeneous coordinates, screw calculus and tensor analysis. the efficiency of any of these methods is very sensitive to the details of the analytical formulation and its numerical implementation. The efficiency also varies with the intended use of the kinematic models.

There are two sides of the same coin describing the kinematics; *forward kinematics* and *inverse kinematics*.

In forward kinematics the joint variables $q_1 \dots, q_6$ are known and the position and orientation of the robot tool are sought.

Inverse kinematics means to compute the joint configuration $q_1, \dots, q_6$ from a given position and orientation of the tool.

The concepts are illustrated in Figure 2.1 [82] and they are shortly described in the following sections.

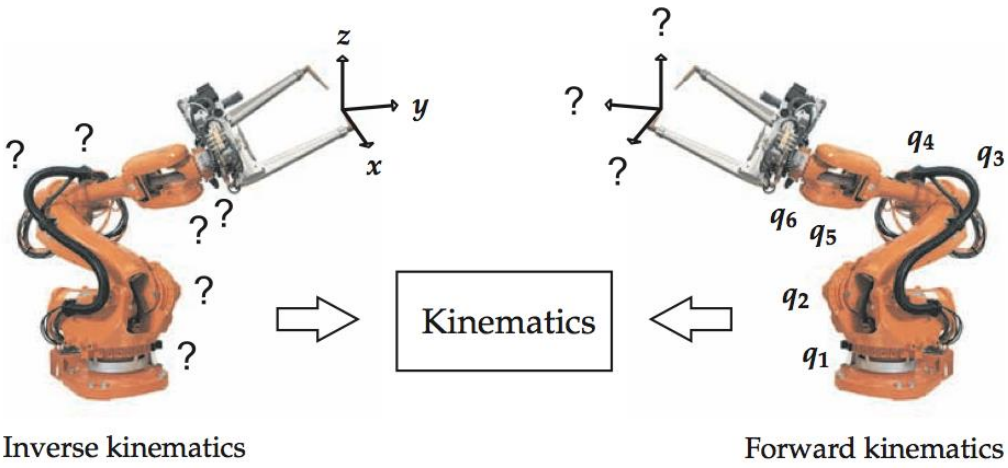


Figure 2.1 : Forward and inverse kinematics.

2.1.4 Forward kinematics

In previous parts the focus is on modeling the forward kinematics. The main interest is the principal structure, and issues regarding efficiently implementation have not been considered. The work is based on homogeneous transformations using the Denavit-Hartenberg (D-H)

2.1.5 Position kinematics

The aim of forward kinematics is to compute the position p and orientation φ of the robot tool as a function of the joint variables q for the robot. By attaching coordinate frames to each rigid body and specify the relationship between these frames geometrically, it is possible to represent the relative position and orientation of one rigid body with respect to other rigid bodies.

2.1.6 Translation

Consider two points; point number i, $p_{0,i}$, and point number j, $p_{0,j}$, expressed in the base coordinate frame 0.A parallel translation of the vector $p_{0,j}$ by the vector d can be described by the relation

$$p_{0,i} = p_{0,j} + \text{d}. \qquad (2.4)$$

The translation is performed in the base frame 0, and d represents the distance and direction from $p_{0,j}$ to $p_{0,i}$.

2.1.7 Rotation

The rotation matrix R_0^1 describes the transformation of the vector p from coordinate frame 1 to frame 0 as

$$p_0 = R_0^1 \, p_1 \qquad (2.5)$$

Where p_1 is the vector of coordinates, expressed in frame 1, and p_0 is the same vector, but expressed in frame 0. The matrix R_0^1 is built upon scalar products between the orthonormal coordinate frames consisting of the standard orthonormal base vectors $\{x_0, y_0, z_0\}$ and $\{x_1, y_1, z_1\}$ in frame 0 and frame 1 respectively. R_0^1 can thus be given by

$$R_0^1 = \begin{pmatrix} x_1x_0 & y_1x_0 & z_1x_0 \\ x_1y_0 & y_1y_0 & z_1y_0 \\ x_1z_0 & y_1z_0 & z_1z_0 \end{pmatrix} \qquad (2.6)$$

Since the scalar product is commutative, the rotation matrix is orthogonal and the inverse transformation $p_1 = R_1^0 \, p_0$ is given by

$$R_1^0 = (R_0^1)^{-1} = (R_0^1)^T \qquad (2.7)$$

Now the coordinate frame 2 is added, related to the previous frame 1 as

$$p_1 = R_1^2 p_2 \qquad (2.8)$$

Combining the rotation matrices in (2.5) – (2.8) gives the transformation of the vector p_2 expressed in frame 2, to the same vector expressed in frame 0 according to

$$p_0 = R_0{}^1\mathrm{R}1 \quad {}^2\mathrm{p}2 = R_0{}^2\mathrm{p}2 \qquad (2.9)$$

The order of the transformation matrices cannot be changed, because $R_0^1 R_1^2$ and $R_1^2 R_0^1$ generally give different results.

In the expressions above the rotations are made around different frames, but sometimes it is desirable to rotate around the fixed frame 0 all the time.

This is performed by multiplying the transformation matrices in the reverse order compared to (2.9),

Giving,

$$R_0^2 = R_1^2 R_0^1 \qquad (2.10)$$

2.1.8 Rigid motion

The most general movement between frame n and frame 0 can be described by a pure rotation combined with a pure translation. This combination is called a *rigid motion* if

$$\mathrm{p_0} = \mathrm{R_0^n P_n} + \mathrm{d_0^n} \qquad (2.11)$$

and the rotation matrix R_0^n is orthogonal, that is, $(R_0^n)^T R_0^n = I$. An important property of the rotation matrix R worth knowing is also that det (R) = 1. The rigid motion can be represented by a matrix of the form

$$\mathrm{H_0^n} = \begin{pmatrix} \mathrm{R_0^n} & \mathrm{d_0^n} \\ 0 & 1 \end{pmatrix} \qquad (2.12)$$

Since R is orthogonal, the inverse transformation is defined by

$$(H_0^n)^{-1} = \begin{pmatrix} (R_0^n)^T & -(R_0^n)^T d_0^n \\ 0 & 1 \end{pmatrix} \qquad (2.13)$$

The transformation is called a *homogeneous transformation*, and it is based on the idea of homogeneous coordinates introduced by Maxwell (1951). The *homogeneous representation* P_i of the vector p_i is defined as

$$P_i = \begin{pmatrix} p_i \\ 1 \end{pmatrix} \qquad (2.14)$$

The transformation can now be written as the homogeneous matrix multiplication

$$P_0 = H_0^1 P_1 \qquad (2.15)$$

Combining two homogeneous transformations

$$\mathrm{p_0} = \mathrm{R_0^1 p_1} + \mathrm{d_0^1} \qquad (2.16)$$

$$p_1 = R_1^2 p_2 + d_1^2 \qquad (2.17)$$

Gives

$$H_0^1 H_1^2 = \begin{pmatrix} R_0^1 & d_0^1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R_1^2 & d_1^2 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} R_0^1 R_1^2 & R_0^1 d_1^2 + d_0^1 \\ 0 & 1 \end{pmatrix} \quad (2.18)$$

$$P_0 = H_0^1 H_1^2 P_2 \quad (2.19)$$

2.1.9 Homogeneous transformations for a robot

The homogeneous matrix representing the position and orientation of frame i relative to frame i − 1,

$$A_i(q_i) = H_{i-1}^i = \begin{pmatrix} R_{i-1}^i & d_{i-1}^i \\ 0 & 1 \end{pmatrix} \quad (2.20)$$

is a function of the single joint variable q_i. It describes the transformation under the assumption that the joints are either revolute or prismatic, The transformation matrix T_i^j that transforms the coordinates of a point expressed in frame j to frame i can then be written as successive transformations as in

$$T_i^j = A_{i+1} A_{i+2} \dots A_{j-1} A_j = \begin{pmatrix} R_i^d & d_i^j \\ 0 & 1 \end{pmatrix} \quad i < j \quad (2.21)$$

Where

$$R_i^j = R_i^{i+1} \dots R_{j-1}^j \quad i < j \quad (2.22)$$

and d_i^j is given recursively by

$$d_i^j = d_i^{j-1} + R_i^{j-1} d_{j-1}^j \quad i < j \quad (2.23)$$

For a robot with n joints, (2.24) gives the homogeneous matrix T_0^n which transforms the coordinates from the tool frame n to the base frame 0 as

$$T_0^n = A_1(q_1) \dots A_n(q_n) = \begin{pmatrix} R_0^n & d_0^n \\ 0 & 1 \end{pmatrix} \quad (2.24)$$

2.1.10 Denavit-Hartenberg representation

In 1955 Denavit and Hartenberg developed a method for describing lower-pair mechanisms (linkages). The idea is to systematically choose coordinate frames for the links. The so-called D-H joint variables represent the relative displacement between adjoining links. The method is commonly used in robotic applications, and

Pieper (1968) and Paul (1977, 1981) were among the first applications to industrial robots. There are two slightly different approaches to the convention, the so-called *standard* D-H notation, described in Spong et al. (2006), and the *modified* D-H form, found in Craig (1989). Both notations represent a joint as two translations and two rotations, but the expressions for the link transformation matrices are quite different.

The D-H link parameters θ_i, a_i, d_i and $\alpha_i$are parameters of link i and joint i, and are defined as follows.

• *Angle* θ_i: angle between the x_{i-1} and x_i axis measured in the plane perpendicular to the z_{i-1}axis.

• *Length* a_i : distance from the origin o_i of frame i to the intersection between the $x_i$and zi−1-axis measured along the xi-axis.

• *Offset* di: distance between the origin o_{i-1} of frame i − 1 and the intersection of the x_i axis with z_{i-1} axis measured along the z_{i-1} axis.

• *Twist* α_i: angle between the z_{i-1} and z_i axis measured in the plane perpendicular to the $x_i$axis.

Table 1.1 shows the DH parameter of Humanoid robot UMAY's arm

2.1.11 Manipulator jacobian

The forward kinematic equations determine the position x and orientation φ of the robot tool given the D-H joint variables q. The *Manipulator Jacobian*, called the *Jacobian* for short, of this function relate the linear and angular velocities v and ω of a point on the robot to the joint velocities $\dot{q}$. The Jacobian is one of the most important quantities in the analysis and control of the robot motion. It is used in many aspects in robot control, like planning and execution of smooth trajectories, determination of singular configurations, execution of coordinated motion, derivation of dynamic equations of motion and to transform tool forces to joint torques.

Generally the n-joint robot has the vector of joint variables $q = (q_1 \dots q_n)^T$. The transformation matrix (2.25) from the tool frame n to the base frame 0 depends on the joint variables q as in

$$T_0^n(q) = \begin{pmatrix} R_0^n(q) & d_0^n(q) \\ 0 & 1 \end{pmatrix} \qquad (2.25)$$

The linear velocity of the robot tool is

$$v_0^n = \dot{d}_0^n \tag{2.26}$$

It can be written on the form

$$v_0^n = J_v \dot{q} \tag{2.27}$$

$$\omega_0^n = J_\omega \dot{q} \tag{2.28}$$

(2.27) and (2.28) can be combined to

$$\begin{pmatrix} v_0^n \\ \omega_0^n \end{pmatrix} = \begin{pmatrix} J_v \\ J_\omega \end{pmatrix} \dot{q} = J_0^n \dot{q} \tag{2.29}$$

Where J_0^n is called the Jacobian. It is a $6 \times n$ matrix because it represents the instantaneous transformation between the n-vector of joint velocities $\dot{q}$ and the 6-vector describing the linear and angular velocities v_0^n, ω_0^n of the robot tool, expressed in the base frame 0.

The Jacobian is an important quantity in robot modeling, analysis and control, since it can tell us about robot characteristics. In this section some of these properties are discussed. A more thorough discussion can be found in any book regarding robot modeling and control [82]

2.2 Classifications Of Visual Servoing Systems

Servo refers to the system that is used to provide control for a device in order to make the output match a desired value. This is achieved with the help of a feedback, which is normally taken form sensors. When the sensors used for feedback are chosen to be the cameras, so-called visual servoing (or vision control) is made possible. Visual servoing systems take a stream of images coming from cameras as input.

In general, the most important task in robotics is the manipulation (e.g. grasping, lifting, and opening) of an object. In order to manipulate an object, it is necessary to interact with the environment and to establish physical contact with the object. A safe and reliable interaction necessitates extensive gathering of information about the environment. Visual servoing methods differ from each other in subject that

mentioned below.

2.2.1 Visual feed forward control

Visual servoing is a term used to describe a closed-loop control of a mechanism (often a robot) by using vision sensors (i.e. cameras). It gives a very accurate positioning result even with bad calibration of the system. It should not be confused with an open-loop control using vision. The visual feed forward control and its difference with feedback control is briefly explained in below.

As discussed, this scheme is sometimes named as look-then-move architecture. This is the most intuitive visually guided system because the object's position is first determined and then the robot manipulator is moved accordingly. This scheme has a high dependency on the accuracy of the image estimation algorithm. In manufacturing, this approach may be naturally appropriate because the visual input only replaces the information about the exact position of the object to handle, which had to be fully characterized before anyway. More developed tasks to generate trajectories including obstacle avoidance may be thus computed. It is common that no visual information about the position of the end-effector is used during motion, but an internal feedback loop which is based on data from each robot link, is employed. In this system, the image information is processed only at the beginning in a sparse and asynchronous way. Its main advantage is that the interpolator is easy to integrate and more complex tasks can be executed.

The main disadvantage of the feed forward design is that it strongly depends on a precise model of the manipulator and of the visual sensor, which includes the camera calibration process. Evidently, strict tolerances in this respect should be taken given that the system deeply relies on a precise calibration. A major constraint is also derived from the image-processing rate, though this also affects other VS architectures [36].

2.2.2 Visual feedback control

The use of visual feedback was initially proposed in the Weiss et al. paper [37] and named as look-and-move algorithm. The main idea is to regulate the error on internal models of the systems by using a continuous visual feedback to sense the position of the robot's end-effector and the object of interest. Visual feedback control is also

known as error-correcting feedback control strategy. In Visual feedback control schemes, later named as Visual Servoing schemes, the outer loop represents the visual registration process which aims to track the object's features in endpoint open loop systems (EOL) or also the robot's end-effector in the case of an endpoint close loop (ECL) schemes. Notice that the output of the controller is the vector of joint velocities, which is later integrated and sent to the joint controller as the controller set-point. Controlling the joints at velocity level provides some advantages and therefore the remaining distance between the object and the manipulator's end-effector is used to compute the desired velocity in the robot actuator, obtaining the velocity of each joint using robot Jacobian matrix.[36]

Also serious problems can be found because the trajectory of the movements is difficult to determine in advance which might cause the target becoming invisible during the course of motion. The object's features might eventually leave the field of view of the camera, generating large errors, which forces the control algorithm to stop. Figure 2.2 shows the Visual Servoing and Open-Loop Control concepts [36].

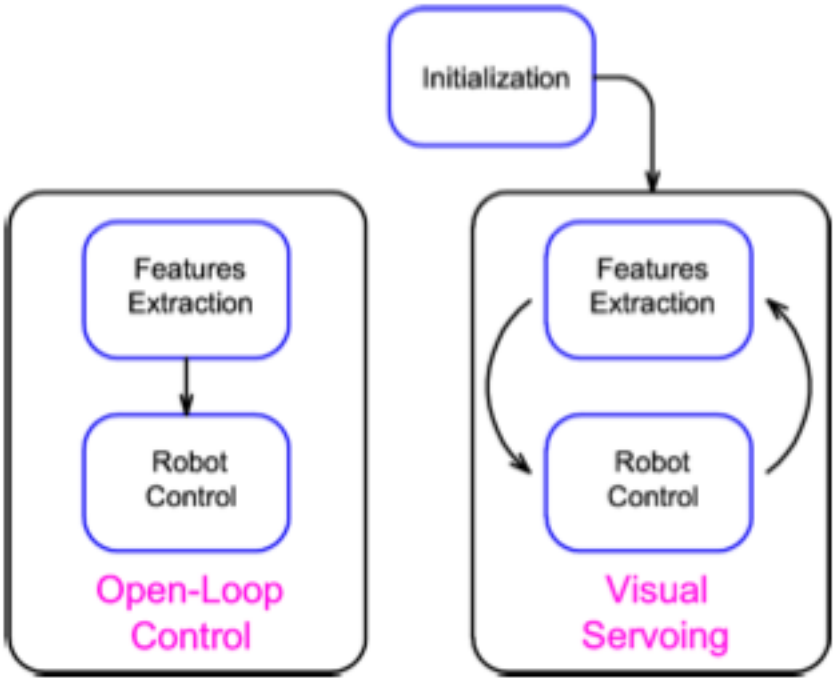


Figure 2.2 : open loop and close loop control.

The more intuitive scheme of visual servoing can be described as follows: by means of an artificial visual algorithm, the object of interest can be detected and marked in the scene. Thus by using background computation, it is possible to calculate the required movements to drive the robotic system to the required location. If such

procedure is repeated for a sequence of required positions of the robot's end- effector with respect to the object of interest then a sequence of required movements for the end-effector to reach such a desired object can be calculated. This primitive scheme was initially known as "look-then-move". It embodies the first attempts to visually guide a robot. Later, as discussed in the next section, the inclusion of a feedback loop was proposed to increase the overall system's accuracy but at a price, it also includes the drawbacks of classic feedback schemes such as higher noise sensitivity and instability risks.

Although a general term of "visual feedback" was first used at the beginning of visually guided models, the more explicative term of "visual servoing" was later used instead. In the early stages, some high specialized hardware and software were required to create a visual servoing system. Currently, after the impressive development in computation and graphic processing speed of an average personal computer, it is possible to design a system capable of driving the robotic arm within real-time constraints. In many published reviews of visual servoing such as [17] and [38], it is discussed that visual servoing was boosted by the arriving of more powerful desktop computing systems which ease the design of VS systems beyond old constraints such as heavy scene analysis computation, slow feature extraction and slow calculation for robot's link demand at a sufficient rate for servoing the manipulator.

Visual servoing requires the direct interaction of image processing, computer vision algorithms, real-time control, robot modelling, control theory and digital signal processing, among others. Therefore, this subject is not an easy topic which requires at least an average knowledge of each of these disciplines. In addition to the difficulty level, a core of real-time software for programming of visual interfaces and real-time robot control are no less than elemental tools in visual servoing.

Given the considerable interaction of several disciplines in VS, it has many common frontiers with other subjects with which it may seem to share task objectives. However, it must be noticed that visual servoing aims to control a robot through artificial vision in a way as to manipulate the environment in contrast with these other disciplines on which the environment is rather observed passively or actively to extract specific information. The control subject has been addressed by many authors especially by Corke in [39].

Last but not least, consider also the relevance of all the image processing algorithms acting in the VS system such as object identification, feature marking, and often reconstruction through artificial vision. All these topics can be discussed under computer vision subject for VS.

Here in, some basic concepts participating in VS scheme is reviewed. The aim of visual servoing is the use of visual information to control the pose (position and orientation) of the final robot actuator or end-effector, with respect to a set of image features from the object of interest or the object's location itself. Thus it is a priority now to discuss how a required robot task can be defined to suit the analysis of the visually guided task.

2.2.3 Camera-robot setup

The design of a visually guided robotic system starts by defining the relationship between the robot's end-effector, the camera and the object of interest. This setup determines how the coordinate systems are assigned and characterizes the required task description. Actually, this is one criterion to classify different visual servoing architectures as discussed below.

Based on the robot-camera configuration, the visual servoing systems are classified as two major classes called *Eye-in-Hand* and *Eye-to-Hand* [40]. In Eye-in-Hand systems, the camera is rigidly mounted on the robot end-effector and in Eye-to-Hand configuration the imaging device is fixed in the workspace to observe both the robot and the object. One or more cameras can be used either as these two configurations or a combination of both. Each of these configurations is used in various servoing tasks based on the limitations of the experiments and applications [41].

Monocular systems use a camera either as a global sensor ("Eye-to-Hand" standalone configuration) or as an Eye–in–Hand configuration. These systems usually adopt some form of model based visual techniques to facilitate the estimation of the depth between the camera and the object [42]. If the Eye-to-Hand configuration is used, a geometric model of the object is commonly used to retrieve the full pose of the object. On the other hand, in the Eye–in–Hand configuration, feature-based tracking techniques are vastly used [43]. A single camera minimizes the processing time needed to extract visual information. However, in the case where the object model is unknown, the loss of depth information limits the servoing operations and

complicates the control design.

In binocular system, two cameras in a stereo configuration can be used to provide complete 3D information about the scene and the object [44]. One of the common approaches is to estimate the disparity, which is then used for depth estimation [45-47]. The fundamental problem of disparity estimation is to match the corresponding features between two or more images [48]. One of the following approaches is usually adopted: i) matching by correlating regions, and ii) matching features (corners, edges) between images. A combination of binocular and Eye-in-hand configuration is seldom used in servoing tasks. Although it may simplify the estimation of the depth, the limited baseline affects the accuracy of the reconstruction.

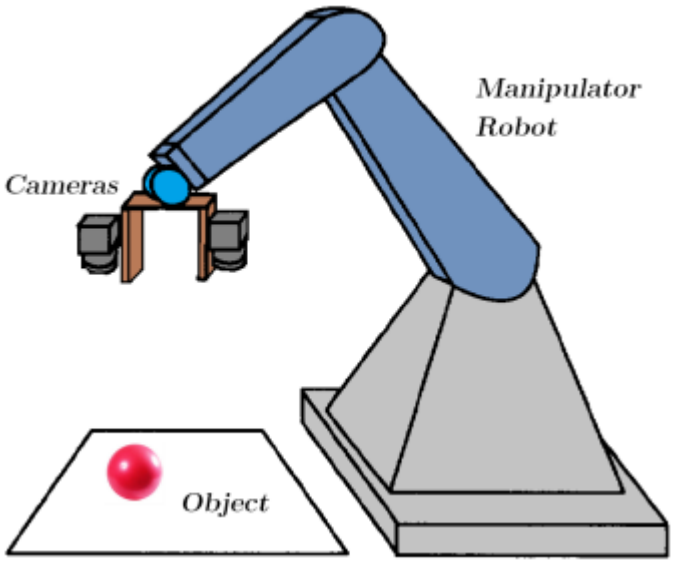


Figure 2.3 : A stereo eye-in-hand system mounted on a manipulator robot's end effector.

Another approach is Binocular, Stand–Alone Configuration; this is a commonly used configuration. Compared to the eye–in–hand stereo con- figuration, it is easy to make the baseline long enough so that the depth estimates are accurate. This approach allows for a wide field of view which makes it easy to observe both the robot and the target simultaneously. Since this method use two different imaging devices, the stereo vision systems require twice as much computational effort in image processing as the monocular systems. [49-51] The multiple cameras provide additional information compared to a single or stereo camera configurations. However, matching across multiple views is usually hard, time consuming and expensive.

Therefore, the research reports on employing more than two cameras for controlling a robot are rare[64].

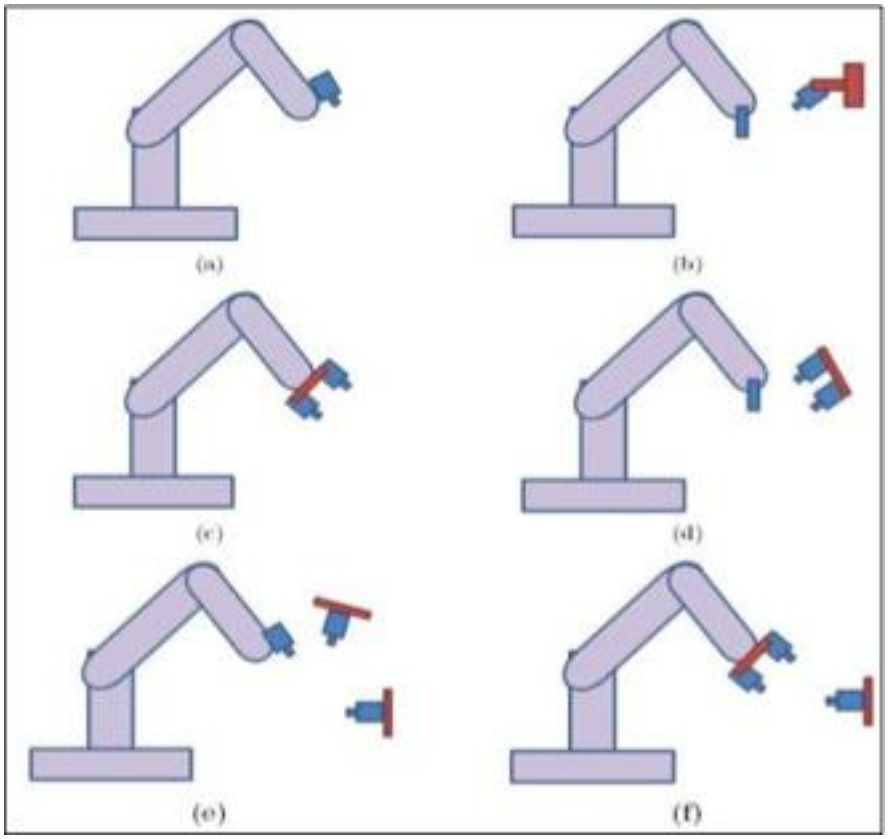


Figure 2.4 : Camera–robot configurations used in visual servoing control.
(a) Monocular eye in hand (b) Monocular Eye to hand

(c) Stereo eye in hand (d) Stereo eye to hand

(e) and (f) combination of eye in hand and eye to hand configuration.

Based on the system error signal, there are two fundamentally different approaches to visual servoing control: Position-Based Visual Servo (PBVS) and Image-Based Visual Servo (IBVS) [52]. Position-based visual servoing (PBVS), shown in Figure 2.5 [59] uses the observed image features from a calibrated camera and a known geometric model of the target to determine the pose of the target with respect to the camera [53]. In a PBVS system, the pose of the target with respect to the camera is estimated [54]. Knowing the target's geometry is essential for the pose estimation problem. In this process, the camera intrinsic parameters and the observed image plane features are needed as well. The robot, then, moves toward that pose and the control is performed in task space. Various algorithms have been proposed for pose estimation [55], [56], it is computationally expensive and relies critically on the

accuracy of the camera calibration and the model of the object's geometry.

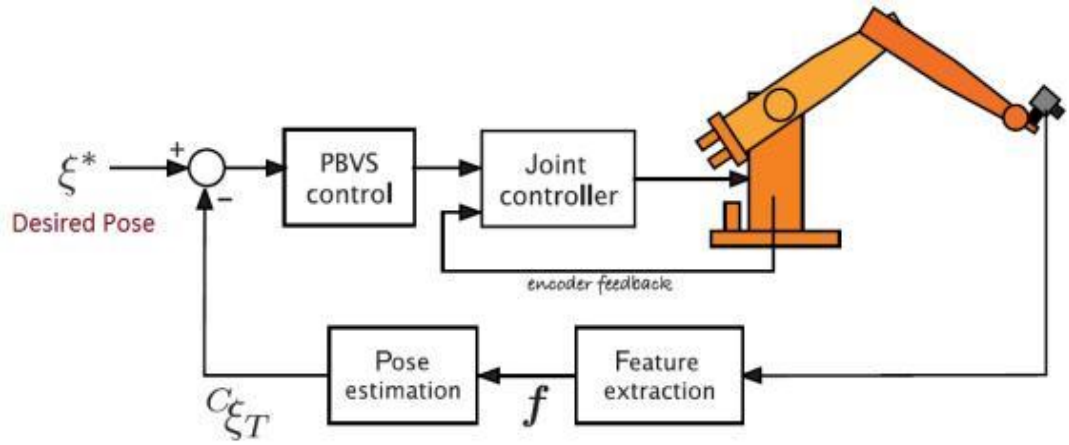


Figure 2.5 : A block diagram of position based visual servo control.

Image-based visual servoing, shown in Figure 2.5 [59], does not include the pose estimation step. It directly uses the image features for control signal calculation. In other words, the control is performed in image coordinate space [57], [58]. The desired camera pose with respect to the target is defined implicitly by the image feature values at the desired pose. The image features are highly non-linear functions of camera pose which make IBVS a challenging control problem. IBVS differs fundamentally from PBVS in omitting the estimation of the relative pose of the target. The relative pose is implicit in the values of the image features. The control problem can be formulated in terms of image coordinates. The task is to move the feature points to the desired position. Moving the feature points in the image space implicitly changes the pose in the Cartesian space.

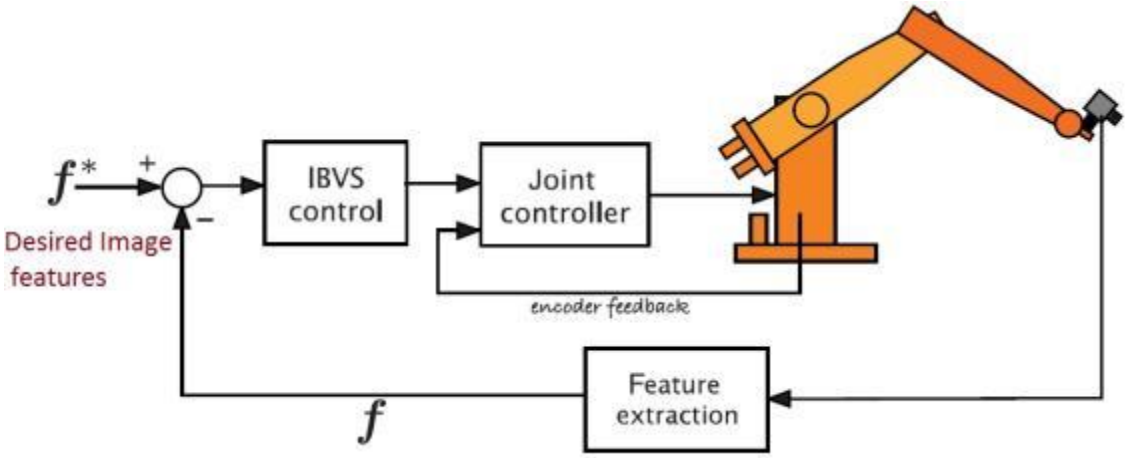


Figure 2.6 : A block diagram of Image based visual servo control.

In a "*Hybrid Visual Servo*" the respective advantages of these schemes are combined

while trying to avoid their shortcomings [60], [61]. The main idea is to use a hybrid of Cartesian and image space sensory feedback signals to control trajectories in both the Cartesian and image spaces simultaneously. One of the drawbacks of this method is that it is more sensitive to image noise than IBVS. Figure 2.7 [59] shows the block diagram of Hybrid visual servo control.

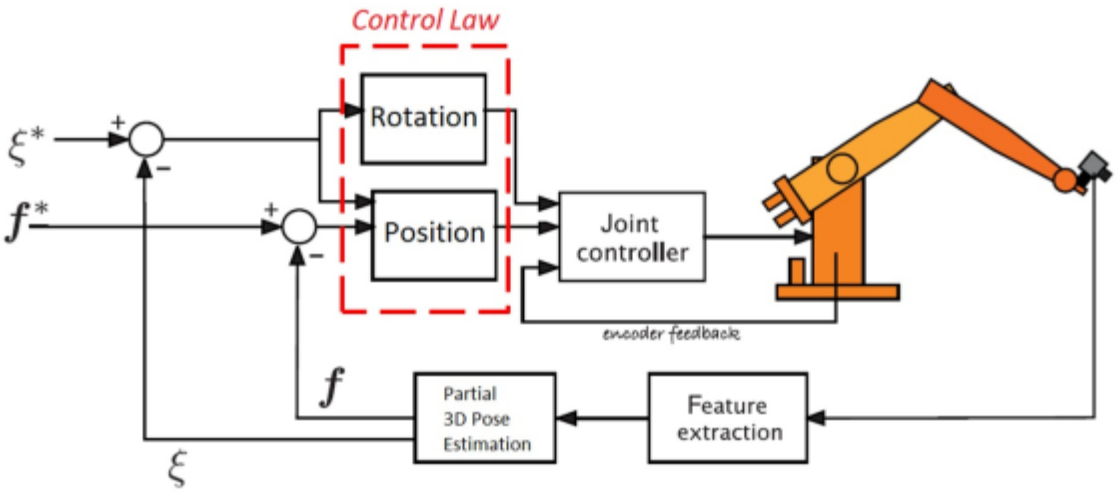


Figure 2.7 : A block diagram of 2-1/2 D visual servoing.

2.2.4 Control issues in visual servoing

It is important at this point to discuss briefly about general control structures used in VS. As presented before, the term Visual Servoing represents a class of dynamic Look-and-move system. In contrast to the direct visual servo control, which directly drives each robot's link torque, a VS system provides the required set points to a joint-level controller. In the literature, nearly all modern VS systems belong to this category, perhaps because the most robots include an integrated joint-level controller. Hence, in the scope of this thesis, visual direct schemes are no longer considered.

2.3 Camera System

The camera used in the robot system is a Kinect camera. The use of Kinect has become widely popular in the robotics community for a myriad of reasons. Some of the main reasons for its rapidly increasing use are its low cost and the fact that it incorporates a depth camera which uses structured IR light to calculate a depth map of the scene being viewed, eliminating the need for stereo systems which tend to be

expensive and require time-consuming calibration procedures. The IR camera can calculate depths with a precision of about 1cm at 2m depth. we should notice that the camera also incorporates a tilt motor which can be used for expanding the field of view.

Table 2.1 : Technical specifications of the Microsoft Kinect camera.

RGB Resolution	640x480 pixels
Depth camera resolution	320x240 pixels
Depth accuracy	11 bits
RGB Framerate	30 fps
Horizontal FOV	57 degrees
Vertical FOV	43 degrees
Tilt unit range	±27 degrees

2.4 Robot Operating System (ROS*)*

ROS is a virtual operating system that works on top of heterogonous computers, it represents an efficient environment for robot software development, it provides hardware abstraction, device drivers, libraries, visualizers, package management. However, the most powerful property is the message-passing between processes, this facilitates the design of complex systems by being divided into small modules with an interaction between them, each module can receive and publish messages, these messages could be sensor readings, control, state, planning, actuator, etc. Messages exchange between different modules is possible even when the modules are on different PCs.

3. VISUAL SERVO CONTROL

In this chapter we are going to talk about the general methods in visual servo control, both position based and image based visual servo control are discussed and interaction matrices in both approach are calculated. The rest of this chapter is organized as follows: a review of classical monocular image based visual servo control (IBVS) will be thoroughly presented. Then, the models of two stereo images of a set of points are built and stereo interaction matrices are derived in "parallel" stereo configurations. The depth information is extracted by the means of epipolar geometry for the interaction matrix calculation. In the final part, the classical monocular IBVS and the proposed stereo IBVS are compared and the advantages and disadvantages of each one with regards to our application will be pointed out.

3.1 Position-Based Visual Servoing Systems (PBVS*)*

Position-based control schemes (PBVS) [26, 29] use the pose of the camera with respect to some reference coordinate frame to define **s**. Computing that pose from a set of measurements in one image necessitates the camera intrinsic parameters and the 3-D model of the object observed to be known. This classical computer vision problem is called the *3- D localization problem* but this problem is beyond the scope of this thesis.

It is convenient to consider three coordinate frames: the current camera frame F_c , the desired camera frame F_{c*} , and a reference frame F_0 attached to the object. We adopt here the standard notation of using a leading superscript to denote the frame with respect to which a set of coordinates is defined.

In PBVS Systems, the features extracted from the image are used to determine the 3D pose X_C of the target, then an error is calculated between the current pose and a desired pose X_d.

the current pose X_C is obtained from the acquired image, then a control function will produce control commands to move the robot from X_C to X_d the target location.Figure 3.1 shows the position-based concept [29]

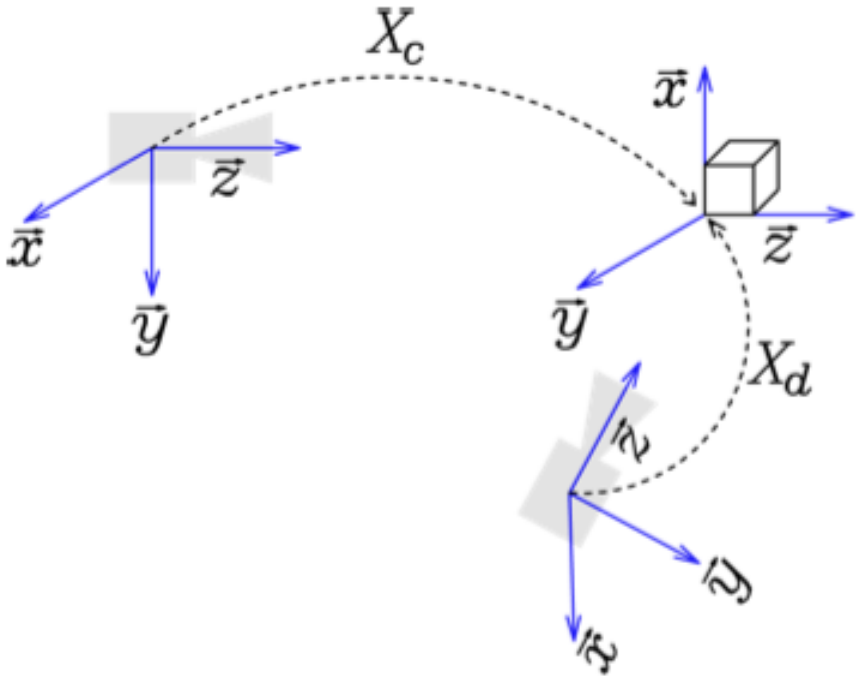


Figure 3.1: Position-Based Visual Control with Eye-In-Hand Camera.

The control commands are the velocity screw of the camera, it could be also converted to the based point of the mobile robot by applying screw transformation. The screw q = [V; Ω] is given as the linear velocity V = [x, y, z] and the angular velocity Ω = [φ,θ,ψ]. Robot controller should transform the velocity screw to the corresponding control signals for joints. Figure 2.4 shows a block diagram for this process [29].

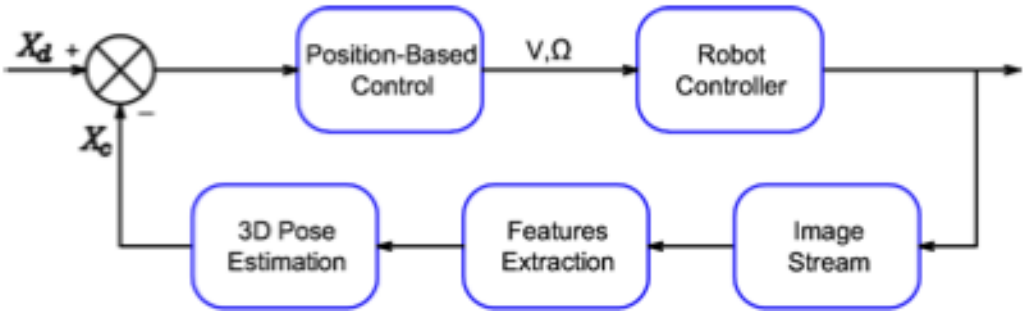


Figure 3.2: Position-Based Visual Control Block Diagram.

This scenario works in the case of static object and moving camera as well as static camera and moving object, the key point is to be able to recover the 3D pose of robot, this will add some constraints on the extracted features and the object itself. Formally, PBVS task is represented as kinematic error function $E : T \rightarrow R^n$ where T is the possible set of positions and orientations that the robot could be configured. The task is said to be achieved for a pose X_C if $E(X_C) = 0$, while for any other configuration the error function will have a non-zero value $E(X_C) = e$, where e is the error vector whose length is less or equal to degrees of freedom of the robot. A control regulator K is used then to reduce such error to zero by producing some desired velocity screw sent to the control, the velocity screw is defined as:

$$\dot{q} \approx Ke \qquad (3.1)$$

In order to define the control and error functions, we have to differentiate between two situations. First when it is possible to recover both target position and orientation [18,19,32,38],and when only position could be extracted [62]. These situations depend on the image processing techniques used as well as the geometric shape of the target. For instance, if the target is a sphere it is not possible to define its pose. In [62], the author discusses the latter case, a point-to-point positioning method is proposed, and the goal is to bring a target point on the robot to a visible fixed point. It controls only 3 DOF of the robot which are the translational velocities. The proposed control law is given by:

$$V_3 = -K(\widehat{x_e}(X_c) - \widehat{x_c}(X_d)) \qquad (3.2)$$

Where $\hat{x}_e$, $\hat{x}_c$ are the estimate of the end-effector and the camera pose respectively. Herein, k is a positive proportional feedback gain. Now, if we move to the general case in which 6 DOF are controlled, we find most of contributions use similar methodology [17,60], and the error function is expressed in terms of image measurements m(t) and camera intrinsic parameters and the 3D model of the object is represented as following:

$$e(t) = s(m(t), a) - s^* \qquad (3.3)$$

Where s is a kind of parameterization of the camera pose and s^* is the desired pose. The coordinate vectors c_{t_0} and $c_{t_0}^*$ give the coordinates of the origin of the object frame expressed relative to the current camera frame and relative to the desired

camera frame, respectively. Furthermore, let $\mathbf{R} = {}^{c^*}\mathbf{R}_c$ be the rotation matrix that gives the orientation of the current camera frame relative to the desired frame.

The parameterization is defined as (t, θ_u), where t is a translation vector and θ_u is the angle/axis rotations as roll, pitch and yaw. By using the object frame as reference, this leads to

$$\begin{cases} s = (c_{t0}, \theta_u) \\ s^* = (c_{t_0}^*, 0), \end{cases} \qquad (3.4)$$

where c_{t0} and $c_{t_0}^*$ are the translation vectors of the base point of the object with respect to the current camera frame and desired frame respectively. Hence the error would be

$$e = \left(c_{t0} - c_{t_0}^* \ , \theta_u\right) \qquad (3.5)$$

The control law used has the same form as 3.1 except that the regulator term k is composed of two terms; one is the inverse of an interaction matrix L_e that relates the error in pose to a corresponding velocity screw, and the other is the gain, it is given by:

$$V_c = -\lambda \, \widehat{L_e^{-1}} e \qquad (3.6)$$

where the interaction matrix (or Jacobian) is basically the time derivative of the camera velocity screw, it is given [60] as:

$$L_e = \begin{bmatrix} -I_3 & [{}^c t_0]_x \\ 0 & L_{\theta u} \end{bmatrix} \qquad (3.7)$$

in which I_3 is the 3 × 3 identity matrix and $L_{\theta u}$ is given by:

$$L_{\theta u} = I_3 - \frac{\theta\,[u]_x}{2} + \left(1 - \frac{sinc\theta}{\frac{sinc^2\theta}{2}}\right)[u]_x^2 \qquad (3.8)$$

Where sinc *x* is the sinus cardinal defined such that *x* sinc *x* = sin *x* and sinc 0 = 1.

Since the dimension *k* of **s** is 6, which is the number of camera degrees of freedom.

By setting:

$$\widehat{L_e^{-1}} = \begin{bmatrix} -I_3 & [{}^{c}t_0]_x L_{\theta u}^{-1} \\ 0 & L_{\theta u}^{-1} \end{bmatrix} \qquad (3.9)$$

Based on 3.6, after some developments, the final control low is given by:

$$\begin{cases} v_c = -\lambda(({}^{c}t_0^* - {}^{c}t_0) + [{}^{c}t_0]_x \theta u) \\ \omega_c = -\lambda \theta u \end{cases} \qquad (3.10)$$

This control law requires the translation and rotation between the object and the camera to be calculated at each iteration, while the gain λ has to be chosen large enough so that it makes the system convergence fast while maintaining the stability. At this point we will not go further with PBVS as it will not be employed later in our solution.

3.2 Image-Based Visual Servoing Systems (IBVS)

This is the most exploited approach compared to position-base and hybrid systems. It is also the one we have chosen to implement based on its advantages and robustness.

In IBVS, the features extracted form the image are directly used in the feedback loop of the servo system. For instance, if the goal is to locate a robot in a specific pose with respect to a target object, this goal is represented by the image taken from the camera in the desired location. The servo task in this case would be, in a simple language, move the robot to a location so the same desired image is obtained, and to be more accurate, the extracted features of the current image have to be equal to the extracted features in the desired pose. Figure 3.3 shows the system block diagrams for IBVS systems. Two main issues have to be analyzed and defined in IBVS approach are as follows:

- The features to be extracted from the image affect the DOF that could be controlled by the robot. In addition, and the method should work accurately in real time.
- The control function that employs the calculated error in producing velocity screw that leads to decrease the error.

This VS class employs image features parameters to calculate the error. No pose estimation algorithms are required because the servoing is performed directly in

image space. A visual servoing task can be represented by an image error function such as e: F $\rightarrow R^m$ with m≤l and l being the dimension of the feature space. The value of such an error function should be zero when the task has been completed.

Figure 3.3 shows the system block diagrams for IBVS systems [29].

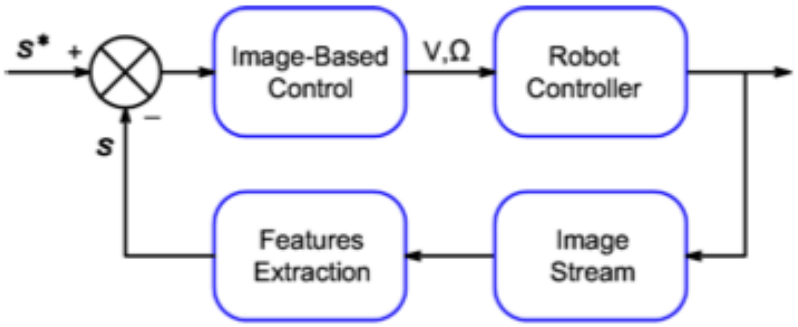


Figure 3.3 : Image based visual servo control.

IBVS has been successfully used in both fixed-camera and Eye-in-Hand setup, as well as in EOL and ECL schemes.

Figure 3.4 shows a very illustrative representation of IBVS as given by Weiss in [24].

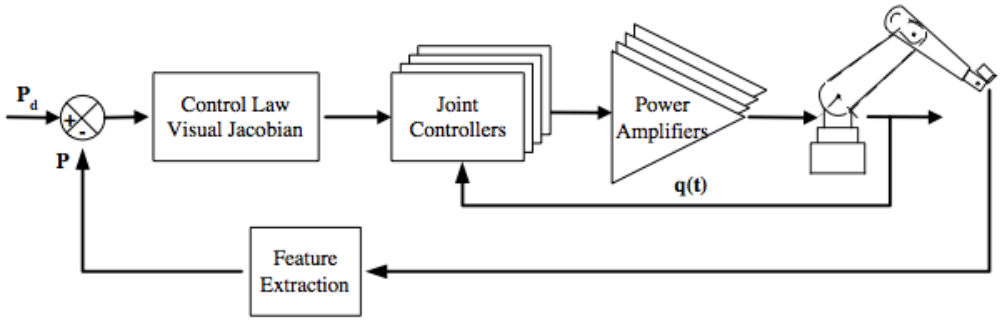


Figure 3.4 : Image-based Visual Servoing blocks representation as introduced by Weiss.

Inside the control law, the visual Jacobian resides in the block which drives the set of joint controllers. Similar to the kinematic Jacobian, the image Jacobian (i.e. visual Jacobian) relates differential changes in the image features to differential changes in the robot position. In general, the main aim of vision-based control schemes is to minimize an error which can be defined by:

$$e(t) = s^* - s \tag{3.11}$$

Where **s** and s^* are vectors of current and desired visual/image features. It is noted that **s** is a function of image measurements (e.g., the image coordinates of interest points or the image coordinates of the centroid of an object) and a set of parameters that represent the same knowledge about the system (e.g., camera intrinsic parameters or the objects 3-D models).

At this point, we consider the case where the target and also the goal pose are fixed, i.e., s^* is constant, and the changes in **s** depend only on camera motion which is controlled with six degrees of freedom (6 DOF), and a camera is attached to the end effector of a six degree-of-freedom manipulator arm.

As it was mentioned before, one of the main classifications of visual servoing schemes is based on system error signal, which is mainly the way how **s** (features) is designed. This classification involves two major approaches: (i) image-based visual servoing control (IBVS), in which **s** consists of a set of features that are immediately available in the image data, (ii) Position-based visual servoing control (PBVS), in which **s** consists of a set of 3-D parameters, which must be estimated from image measurements. After selecting a proper feature set for **s,** we need to design a control scheme which is a velocity controller. To design such controller, we require the relationship between the time variation of **s** and the camera velocity:

$$\dot{s} = J_s u_c \tag{3.12}$$

Where $u_c = (v_c, \omega_c)$ is the spatial velocity of the camera, v_c is the instantaneous linear velocity of the origin of the camera frame, ω_c is the instantaneous angular velocity of the camera frame, $J_s \in R^{k\times 6}$ is called *Jacobian Matrix* or *feature interaction matrix*. Since $s^*$is constant, its time derivatives is equal to zero. Consequently we have:

$$\frac{ds^*}{dt} = 0$$

$$\rightarrow \frac{d}{dt} e(t) = \frac{d}{dt}(s - s^*) = \frac{ds}{dt} = \dot{s} \tag{3.13}$$

Then we will have:

$$e^{\cdot} = J_e \; u_c \qquad (3.14)$$

, Where $J_e = J_s$. A traditional proportional controller u_c would be the input to the robot controller. By Letting $e^{\cdot} = -\lambda \, e$, The control signal can be designed as:

$$\mathrm{u_c} = -\lambda \, J_s^{+} \, e \qquad (3.15)$$

The interaction matrix J_s depends on the chosen features, and it relates the velocity of the objects in 3D world to the velocities of the projection (or features) of such objects in the image. Common features to be extracted are points, lines, segments, spheres, eclipses.

In Figure 3.5 [8], the pixel coordinates of a point in the image is considered as a feature of interest since it is easy to detect, and also many objects could be expressed as points.

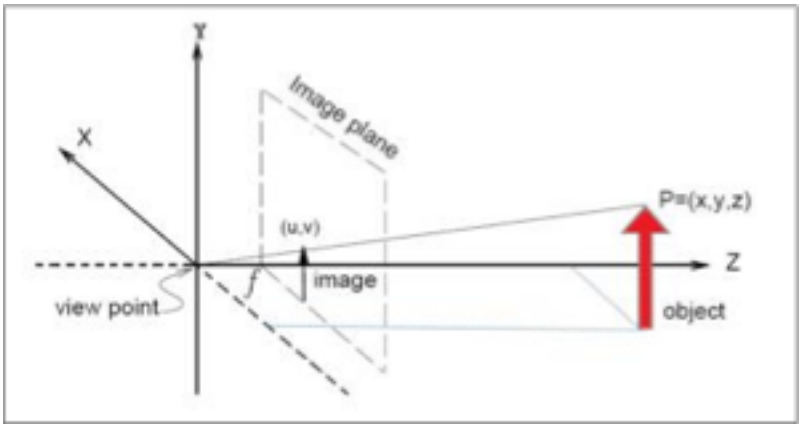


Figure 3.5 : pixel coordinate of point in image.

In more details, if X= (X, Y, Z) is a 3D point in the space expressed in camera coordinate system, and x= (x, y) is the projection of that point in the image. The relationship between both points is given by a perspective projection equation:

$$\begin{cases} x = \dfrac{X}{Z} = \dfrac{u - u_0}{f} \\ y = \dfrac{Y}{Z} = \dfrac{v - v_0}{f} \end{cases} \qquad (3.16)$$

Where (u, v) are the coordinates of the point in pixels, (u_0, v_0) is the principal point and f is the focal length. Now, if s=x= (x, y) is taken as selected feature, to relate the velocity of the 3D point X to the velocity of its projection x, the time derivative of 3.16 is taken, we obtain:

$$\begin{cases} x^{\cdot} = \dfrac{X^{\cdot} - xZ^{\cdot}}{Z} \\ y^{\cdot} = \dfrac{Y^{\cdot} - yZ^{\cdot}}{Z} \end{cases} \quad (3.17)$$

Then we can obtain:

$$\begin{cases} X^{\cdot} = -v_x - \omega_y Z + \omega_z Y \\ Y^{\cdot} = -v_y - \omega_z X + \omega_x Z \\ Z^{\cdot} = -v_z - \omega_x Y + \omega_y X \end{cases} \quad (3.18)$$

Therefore, the time derivative of X could be obtained from the well known optical flow equation:

$$X^{\cdot} = -v_C - \omega_C \times X \quad (3.19)$$

Where $v_c = [v_x, v_y, v_z]^T$,$\omega_c = [\omega_x, \omega_y, \omega_z]^T$are the linear and the angular velocities, respectively. Now by using 3.18, 3.19 we can extract the interaction matrix as following:

$$x^{\cdot} = -\frac{v_x}{Z} + \frac{xv_z}{Z} + xy\omega_x - (1 + x^2)\,\omega_y + y\omega_z \quad (3.20)$$

$$y^{\cdot} = -\frac{v_y}{Z} + \frac{yv_z}{Z} + (1 + y^2)\omega_x - xy\omega_x - x\omega_z \quad (3.21)$$

Which can be written:

$$x^{\cdot} = J_x\, v_c \quad (3.22)$$

Where the interaction matrix L_x related to x is:

$$J_x = \begin{bmatrix} -1/Z & 0 & x/Z & xy & -(1 + x^2) & y \\ 0 & -1/Z & y/Z & 1 + y^2 & -xy & -x \end{bmatrix} \quad (3.23)$$

In the matrix J_x , the value Z is the depth of the point relative to the camera frame. Therefore, any control scheme that uses this form of the interaction matrix must estimate or approximate the value of depth which is normally time-varying and unknown especially in a monocular camera vision system. After the interaction matrix is obtained, forming the control law is now straight forward, replace $x^{\cdot}$ by the error $s^{\cdot}$ and multiply both sides by the pseudo inverse of the interaction matrix, then adding some gain λ, the control signal can be designed as:

$$v_c = -\lambda J_x^{+} e \qquad (3.24)$$

but we should notice that where $J_x^{+} \in R^{6\times k}$ is the Moore-Penrose pseudo-inverse of J_x that is

$$J_x{}^{+} = (J_x{}^{T} J_x)^{-1} J_x{}^{T} \qquad (3.25)$$

When J_e is of full rank 6. And when k = 6 and if det $(J_x) \neq 0$ we can obtain:

$$v_c = -\lambda J_x^{-1} e \qquad (3.26)$$

Otherwise, when it is not a full rank, an approximation or an estimation of it can be determined; we obtain the final control law:

$$v_c = -\lambda \widehat{L_e^{+}} \dot{s} \qquad (3.27)$$

Two questions arise at this step, (i) how many degrees of freedom could be controlled by this method, and (ii) how many points are required as features.

The DOF to be controlled have to be equal or less than the rank of the interaction matrix, hence, to control 6 DOF, we need at least 3 points to form (6 × 6) interaction matrix. It is formed by the concatenation of the interaction matrices for each point as in 3.23 The three points should not lie on the same line, else the rank will be less than 6. This rank condition is also true from a logical point of view.

As it was said, to control a 6 DOF system, we need at least three points ($k \geq 6$). In this case, there may be some configurations of the system making **Lx** singular. Therefore, we need to consider more than three feature points in the feature vector. If we use the feature vector including four points in image plane, $s = (p_1 , p_2 , p_3 , p_4)$ the Interaction or interaction matrices for four points would be:

$$L_x = \begin{bmatrix} L_{x1} \\ L_{x2} \\ L_{x3} \\ L_{x4} \end{bmatrix} \qquad (3.28)$$

Where $L_x \in R^{8\times6}$, $L_{xi} \in R^{2\times6}$, i=1,2…,4.

Similarly, the camera intrinsic parameters are involved in the computation of x and

y.

An important issue is met in IBVS algorithm during the calculation of the interaction matrix in practice. In literature, three main ways to estimate the interaction matrix are more prominent:

(i) Interaction matrix calculated from the current pose, it requires depth estimation. For instance, in point features interaction matrix 3.23 the depth Z of the point has to be defined. [42]

(ii) Calculated at the desired pose and stays constant, this means less computation since it is constructed once and does not need depth estimation. [42]

(iii) Hybrid estimation [57] by taking the average of the previous two matrices, produce less velocity oscillations compared to each matrix individually. However, it requires the depth information as well.

In the following chapters, the details of these methods will be discussed in a more elaborate way.

3.3 Interaction Matrix For Image-Based Stereo Visual Servoing

Having all the necessary information from classical monocular IBVS, the approach and control scheme will now be extended to a stereo image-based visual servoing. Stereo vision is a technique that uses two cameras to measure the 3D distances from the cameras, similar to human depth perception with human eyes. The process uses two parallel cameras aligned at a known distance of separation. Each camera captures an image and these images are analyzed for common features. Triangulation is used with the relative position of these matched pixels in the images.[51] Also there is another format of stereo vision in which cameras are not parallel.

In the first case, we consider our stereo-vision model composed of two parallel cameras which are perpendicular to the baseline. The focal points of two cameras are apart by the distance *b/2* with respect to origin of sensor frame {C} on the base line which means the origin of the camera frame is in the center of these points. Focal distance of both cameras is *f;* therefore, the image planes and corresponding frames for left and right cameras are located with the distance *f* from the focal points and orthogonal to the optical axis. We assign {L} and {R} as the frames of the left and

right images.

Figure 3.6 illustrates the case where both cameras observe a 3D point ${}^{C}P$ [25].

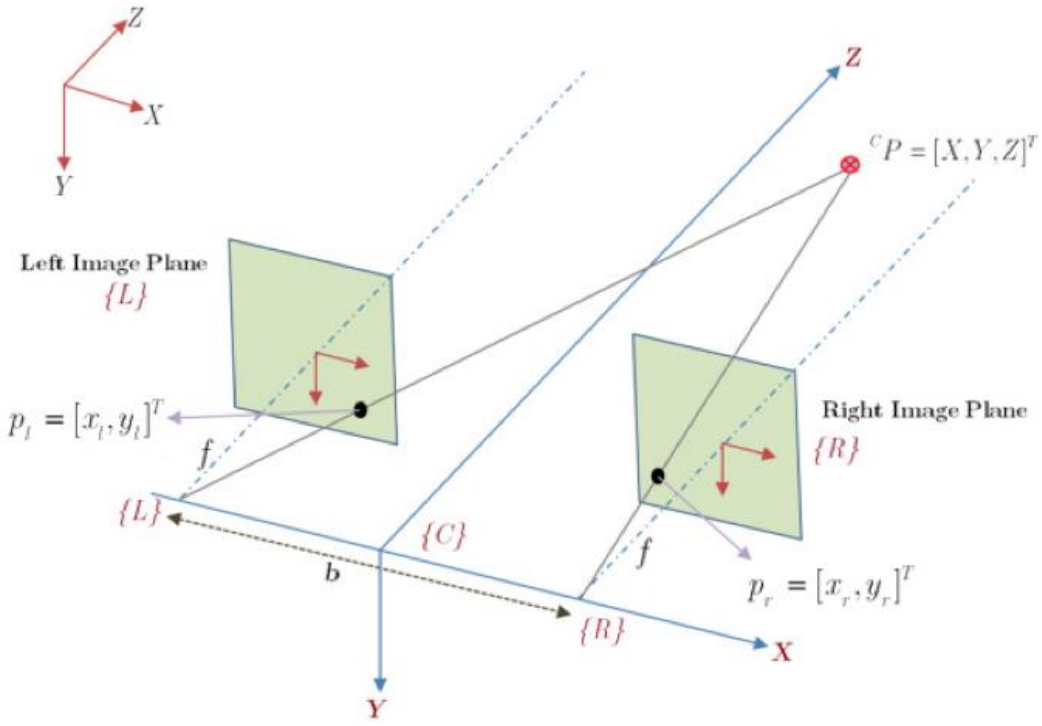


Figure 3.6 : Model of the stereo vision system observing a 3D point.

The feature vector is defined as:

$$s = \begin{bmatrix} x_l \\ y_l \\ x_r \\ y_r \end{bmatrix} \tag{3.29}$$

Where $p_r = [x_l, y_l]^T$, $p_r = [x_r, y_r]^T$ are the image coordinates of the 3D point, observed by the left and right cameras respectively. Now we can use the following equations (3.30−3.33) to project observed point into left and right image planes;

$$x_l = \frac{X + b/2}{Z} = \frac{(u_l - \sigma_u)}{f^* \alpha} \tag{3.30}$$

$$y_l = \frac{Y}{Z} = \frac{(v_l - \sigma_v)}{f^*} \tag{3.31}$$

$$x_r = \frac{X - b/2}{Z} = \frac{(u_r - \sigma_u)}{f^* \alpha} \tag{3.32}$$

$$y_r = \frac{Y}{Z} = \frac{(v_r - \sigma_v)}{f^*} \tag{3.33}$$

where $p_l = [x_l, y_l]^T$, $p_r = [x_r, y_r]^T$ are the normalized coordinates from $[u_l, v_l]^T$, $[u_r, v_r]^T$, and f^* is focal length described in pixel dimensions.$(\sigma_u, \sigma_v, f, f^*, \alpha)$ is the set of camera intrinsic parameters: σ_u and σ_v are the coordinates of the camera principal point, f is the focal length, and α is the ratio of the pixel dimensions where $dy/dx = \alpha$.

Taking the time derivative of the perspective projection equations, we can obtain:

$$\dot{x_l} = \frac{\dot{X}}{Z} - \frac{(X + b/2)\dot{Z}}{Z^2} = \frac{(\dot{X} - x_l \dot{Z})}{Z} \quad (3.34)$$

$$\dot{y_l} = \frac{\dot{Y}}{Z} - \frac{Y\dot{Z}}{Z^2} = \frac{(\dot{Y} - y_l \dot{Z})}{Z} \quad (3.35)$$

$$\dot{x_r} = \frac{\dot{X}}{Z} - \frac{(X + b/2)\dot{Z}}{Z^2} = \frac{(\dot{X} - x_r \dot{Z})}{Z} \quad (3.36)$$

$$\dot{y_r} = \frac{\dot{Y}}{Z} - \frac{Y\dot{Z}}{Z^2} = \frac{(\dot{Y} - y_r \dot{Z})}{Z} \quad (3.37)$$

The relation between a velocity of a feature point in an image p_l and a velocity of a feature point in a camera frame ${}^{C}P$ is given as

$$\dot{p_I} = {}^{I}J_{C}\,{}^{C}P \quad (3.38)$$

$$p_I = [p_l, p_r]^T = [x_l, y_l, x_r, y_r]^T \quad (3.39)$$

$$J_c = \begin{bmatrix} \frac{\partial x_l}{\partial X} & \frac{\partial x_l}{\partial Y} & \frac{\partial x_l}{\partial Z} \\ \frac{\partial y_l}{\partial X} & \frac{\partial y_l}{\partial Y} & \frac{\partial y_l}{\partial Z} \\ \frac{\partial x_r}{\partial X} & \frac{\partial x_r}{\partial Y} & \frac{\partial x_r}{\partial Z} \\ \frac{\partial y_r}{\partial X} & \frac{\partial y_r}{\partial Y} & \frac{\partial y_r}{\partial Z} \end{bmatrix} \quad (3.40)$$

Consequently, using Equation (3.20) in (3.23) we obtain;

$$J_c = \begin{bmatrix} \frac{1}{Z} & 0 & -\frac{X + b/2}{Z^2} \\ 0 & \frac{1}{Z} & -\frac{Y}{Z^2} \\ \frac{1}{Z} & 0 & -\frac{X - b/2}{Z^2} \\ 0 & \frac{1}{Z} & -\frac{Y}{Z^2} \end{bmatrix} \quad (3.41)$$

As it is mentioned before, the velocity of ${}^{C}P = [X, Y, Z]^T$ can be related to the camera spatial velocity using the equation:

$$P^{\cdot} = -\omega_c \times {}^{C}P - v_c \qquad (3.42)$$

Therefore, we can obtain;

$$\dot{P} = \begin{bmatrix} -\omega_y Z + \omega_x Y - v_x \\ -\omega_z X + \omega_x Z - v_y \\ -\omega_x Y + \omega_y X - v_z \end{bmatrix} \qquad (3.43)$$

$$= \begin{bmatrix} -1 & 0 & 0 & 0 & -Z & Y \\ 0 & -1 & 0 & Z & 0 & -X \\ 0 & 0 & -1 & -Y & X & 0 \end{bmatrix} \begin{bmatrix} v_c \\ \omega_c \end{bmatrix} \qquad (3.43)$$

Substituting Equation (3.18) in (3.23) we have:

$$\dot{P}_I = {}^{I}J_C \, {}^{C}P^{\cdot} \qquad (3.44)$$

$$\dot{P}_I = {}^{I}J_C \begin{bmatrix} -1 & 0 & 0 & 0 & -Z & Y \\ 0 & -1 & 0 & Z & 0 & -X \\ 0 & 0 & -1 & -Y & X & 0 \end{bmatrix} u_c \qquad (3.45)$$

$$= J_{st} u_c$$

Where J_{st} is the stereo-vision image interaction which expresses the relation between a velocity of a feature point in an image, $\dot{P}_I = [\dot{x}_l, \dot{y}_l, \dot{x}_r, \dot{y}_r]^T$ and a moving velocity u_c of a camera;

$$J_{st} = \begin{bmatrix} \frac{-1}{Z} & 0 & \frac{X + b/2}{Z^2} & Y\frac{X + b/2}{Z^2} & -(1 + X\frac{X + b/2}{Z^2}) & \frac{Y}{Z} \\ 0 & \frac{-1}{Z} & \frac{Y}{Z^2} & 1 + (\frac{Y}{Z})^2 & -\frac{XY}{Z^2} & -\frac{X}{Z} \\ \frac{-1}{Z} & 0 & \frac{X - b/2}{Z^2} & Y\frac{X - b/2}{Z^2} & -(1 + X\frac{X - b/2}{Z^2}) & \frac{Y}{Z} \\ 0 & \frac{-1}{Z} & \frac{Y}{Z^2} & 1 + (\frac{Y}{Z})^2 & -\frac{XY}{Z^2} & -\frac{X}{Z} \end{bmatrix} \qquad (3.46)$$

From the model of the stereo vision projection Equations (3.18),(3.20) the following equations hold for 3D coordinates of the observed point;

$$X = \frac{b}{2}\frac{x_l + x_r}{x_l + x_r} \qquad (3.47)$$

$$Y = y_l \frac{b}{x_l - x_r} = y_r \frac{b}{x_l - x_r} \qquad (3.48)$$

$$Z = \frac{b}{x_l - x_r} \qquad (3.49)$$

Therefore, we can rewrite the stereo-vision image interaction matrix as:

$$J_{st} = \begin{bmatrix} \frac{-a}{b} & 0 & x_l\frac{a}{b} & x_l y & -(1+\frac{x_l(x_l+x_r)}{2}) & y \\ 0 & \frac{-a}{b} & y\frac{a}{b} & 1+y^2 & -y\frac{(x_l+x_r)}{2} & -\frac{(x_l+x_r)}{2} \\ \frac{-a}{b} & 0 & x_r\frac{a}{b} & x_r y & -(1+\frac{x_r(x_l+x_r)}{2}) & y \\ 0 & \frac{-a}{b} & y\frac{a}{b} & 1+y^2 & -y\frac{(x_l+x_r)}{2} & -\frac{(x_l+x_r)}{2} \end{bmatrix} \qquad (3.50)$$

When we use n feature points, interaction matrices $J_{st1}, \dots J_{stn}$ are given from the coordinates of each feature points in an image.[63] Therefore in case of n-feature points we can express the image interaction matrix as:

$$J_{st} = \begin{bmatrix} J_{st1} \\ \vdots \\ J_{stn} \end{bmatrix} \qquad (3.51)$$

as it was mentioned there is an other case when cameras are not parallel, that is more complicated than this method and hardly was used so it is not explained .

3.4 Discussion

After we have reviewed most of the methods, which are relevant for our context, in this section we are going to analyze the advantages and disadvantages for each category in order to make a decision about a suitable method.

Early visual servo systems relied on perfect calibration of the robot/camera system and mainly adopted position based visual servoing. The tasks were performed in structured and controlled environments. The position based approach is nowadays mostly used in connection with trajectory generation for obstacle avoidance or tracking of a moving target.

If we start by the PBVS, we could list the following disadvantages:

- It requires the geometrical model of the target that could be unavailable or has a complex shape that makes the modeling difficult.

- In order to achieve an accurate positioning, a camera calibration is mandatory, this makes the computed quantities sensitive to calibration error.
- The possibility of estimating the 3D pose is target's shape dependent.
- Acquiring the 3D pose of the robot with respect to the target is computationally expensive [43] as it includes two computations of least squares for optimizations. We could list the following disadvantages/requirements of IBVS :
- The conventional IBVS requires that the target to be expressed as geometrical primitives such as points, lines, circles, etc. Furthermore, to control a certain number of DOF, it requires a minimum number of selected features.
- Selected image features have to be traceable among the image sequence.
- The trajectory performed by the end-effector is not predicted, not controllable, not linear and not optimal.

As soon as the first two issues are satisfied, and the trajectory is not important, the IBVS has better performance than PBVS in terms of sensitivity to measurements and calibration errors, computation cost and stability. Furthermore, it is more appropriate for feature-less objects. For instance, if we have one trackable point, we can control 2 DOF in IBVS while the PBVS will not work as it needs minimum 3 points to recover the robot pose. The $2\frac{1}{2}$D visual servo suffers also from the same problem. In terms of popularity, PBVS has been less adopted, compared to IBVS, by researchers in the last decade except when complementary sensors are used. As the current trend in robotics is toward the capability of a robotic system to operate in highly dynamic (changing) environments. Also Position based systems require calibration which may be difficult, time consuming or even impossible to obtain. Image based visual servoing is more adequate and commonly used in cases where accurate calibration parameters are not known.

The visual information can be obtained through monocular vision (one camera), stereo vision (two cameras) or a redundant system that uses several cameras, typically three or four.

One of the drawbacks of image based systems is the computation of the image Jacobian. As already mentioned, it depends on the distance between the camera and

the target which is in direct relation to the camera configuration used. Many monocular systems utilize a constant Jacobian or perform a partial pose estimation which requires some pre-knowledge about the target shape. This, of course, greatly affects the flexibility of the servo system.

If we just make a rough comparison between monocular and stereo systems, it was clear that it is time consuming. The reason is that the former one processes one image in monocular approach in each iteration whereas the latter processes two in stereo per iteration.

Therefore, as stated in many works (REF), the first one, monocular vision, is the most popular because it is the cheapest and the most viable approach to run in real time. However, with only one camera, the problem is that we cannot measure the depth. Stereo vision solves the depth measurement problem, but it introduces complexity in the algorithmic processing, which may not be tolerable in real time systems. Several cameras are normally just used when a high accuracy is required and there is enough time for processing.

Considering the existing visual servo systems, it seems that the general trend is to concentrate on one part of the whole servoing loop. The focus is either on developing a fast and reliable perception part of the system or demonstrating a new and flexible control design. These two issues are not independent and both of them should be considered in order to design a robust and flexible visual servo system.

In next chapter, an image-based visual servoing approach based on depth camera such as ‘Kinect’ is presented and mathematical discussion is given.

In this work, visual servo control (IBVS scheme) by using depth camera in both eye-in-hand and eye-to-hand configuration is introduced.

4. TIME-OF-FLIGHT AND KINECT IMAGING

The acquisition of the geometry description of a dynamic scene has always been a very challenging task, which required state-of-the-art technology and instrumentation only affordable by research labs or major companies until not too long ago. The recent arrival on the market of matricial Time-of-Flight range cameras (or simply *ToF cameras*) and the even more recent introduction of the Microsoft Kinect range camera (or simply *Kinect* in the sequel), accompanied by massive sales of this product, has made widely available depth data streams at video rate of generic still or dynamic scenes. A truly amazing possibility considering that until now streaming at such rates was possible only with standard image sequences. In the computer graphics, computer vision and image processing communities a large interest arose for these devices and questions of the following kind have become common: "What is a ToF camera?", "How does the Kinect work?" Although ToF cameras and Kinect as depth cameras, i.e, as providers of depth data, are functionally equivalent, it is also important to remember that there exist fundamental technological differences between them, which cannot be ignored. Figure 4.1 briefly shows the Depth measurement Techniques [65].

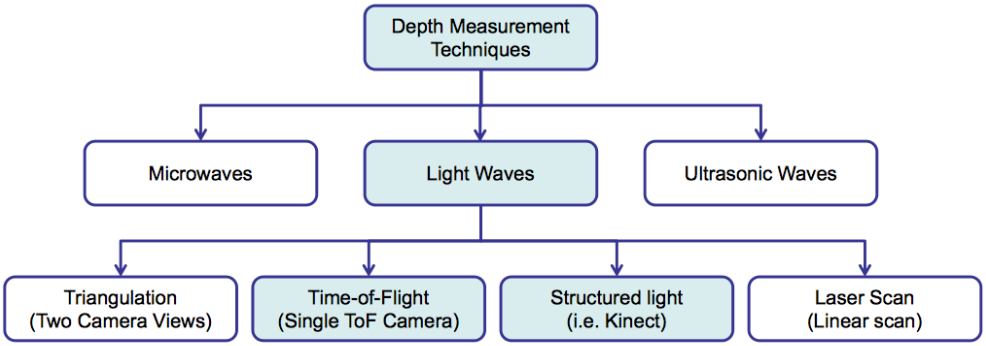


Figure 4.1: Depth measurment techniques.

4.1 Applications For 3D Sensing

it can be classifeid as below:

- Computer Vision
  - People and object tracking
  - 3D Scene reconstruction
- Interaction
  - Gesture-based user interfaces
  - Gaming/character animation

- Medical
  - Respiratory gating
  - Ambulatory motion analysis

4.1.2 Depth measurement using multiple camera views

As It was mentioned in previous chapters ,one of the common way to get Depth Measurement, is Using Multiple Camera Views . Figure 4.2 shows the Multiple camera pictures [65].

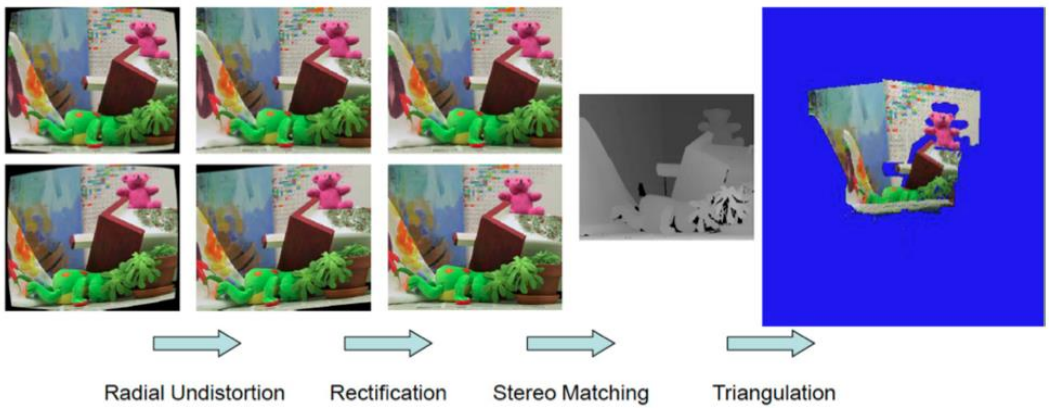


Figure 4.2 : Multiple camera pictures.

which has disadvantages that can be classified as:

- At least two calibrated cameras required
- Multiple computationally expensive steps
- Dependence on scene illumination
- Dependence on surface texturing

4.2 Classification Of Depth Measurement Techniques

The synopsis of distance measurement methods is shown on Figure 4.3 which was derived from [65], It offers a good framework to introduce such differences between. For the purposes of this work it suffices to note that the reflective optical methods of Figure 4.3 are typically classified into *passive* and *active*. Passive range sensing refers to 3D distance measurement by way of radiation (typically, but not necessarily, in the visible spectrum) already present in the scene, and stereo-vision systems are a classical example of this family of methods; active sensing refers, instead, to 3D distance measurement obtained by projecting in the scene some form of radiation as made, for instance, by ToF cameras and by light coding systems of which the Kinect is a special case.

The operation of ToF cameras and Kinect involves a number of different concepts about ToF sensors, imaging systems and computer vision. The depth or distance measurements taken by the systems of Figure 4.3 can be typically converted into depth maps, i.e. [66].

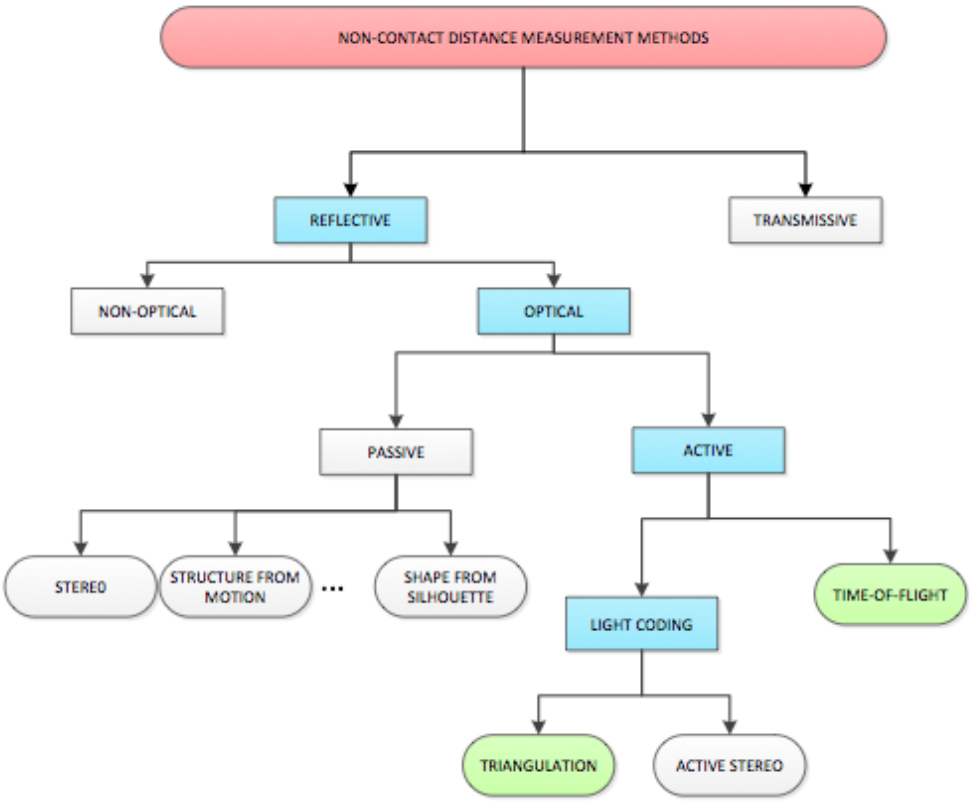


Figure 4.3 : Taxonomy of distance measurement methods.

data with each spatial coordinate (*x*, *y*) associated to depth information *z*, and the depth maps can be combined into full all-around 3D models.Time-of-Flight (ToF) cameras produce a *depth image*, each pixel of which encodes the distance to the corresponding point in the scene.

These cameras can be used to estimate 3D structure directly, without the help of traditional computer-vision algorithms. There are many practical applications for this new sensing modality, including robot navigation [67], 3D reconstruction [68] and human-machine interaction [71].

ToF cameras work by measuring the phase-delay of reflected infrared (IR) light. This is not the only way to estimate depth; for example, an IR *structured-light* pattern can be projected onto the scene, in order to facilitate visual triangulation [69]. Devices of this type, such as the Kinect [70], share many applications with ToF cameras. Figure 4.4 and 4.5 show regular and depth image respectivly [70].

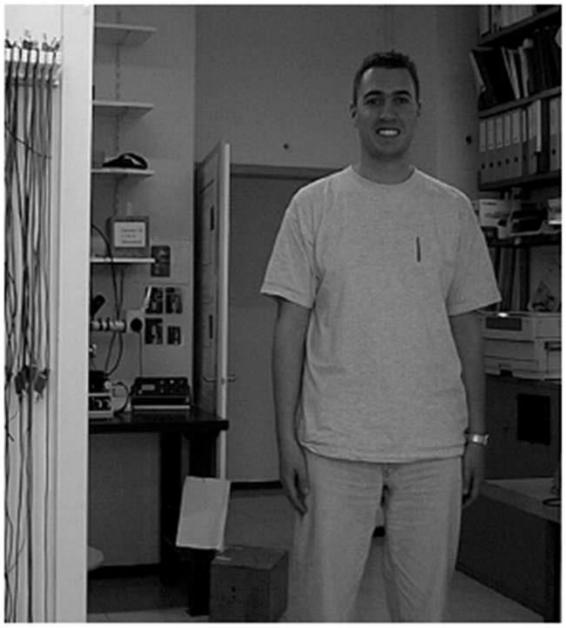


Figure 4.4 : Regular Camera Image.

Actually Time-of-Flight (ToF) Imaging refers to the process of measuring the depth of a scene by quantifying the changes that an emitted light signal encounters when it bounces back from objects in a scene.

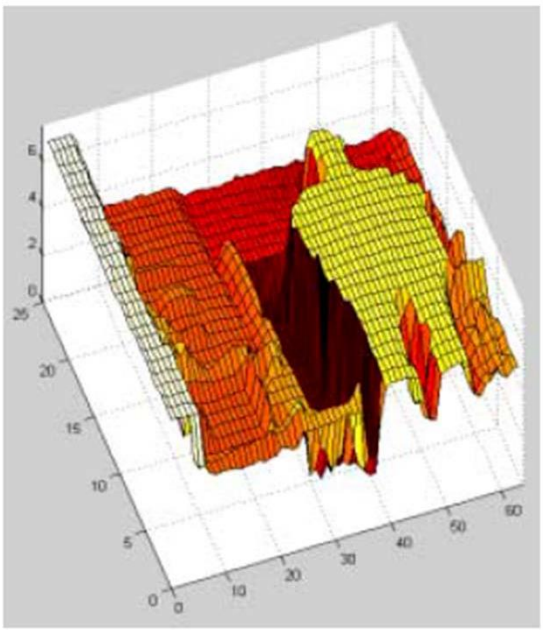


Figure 4.5 : ToF Camera Depth Image.

4.2.1 Advantage of depth measurement using a ToF camera

- Only one (specific) camera required
- No manual depth computation required
- Acquisition of 3D scene geometry in real-time
- Reduced dependence on scene illumination
- Almost no dependence on surface texturing

4.3 Basics Of ToF Sensors

A point-wise ToF sensor estimates its *radial* distance from a scene point by the Time-of-Flight or RADAR (Radio Detection And Ranging) principle.

In simple words, since the electro-magnetic radiation travels in the air at light speed

$c \approx 3 \times 10^8$ [m/s] , the distance ρ covered at time τ by an optical radiation is $\rho = c\tau$. Figure 4.6 shows the typical ToF measurement scheme:

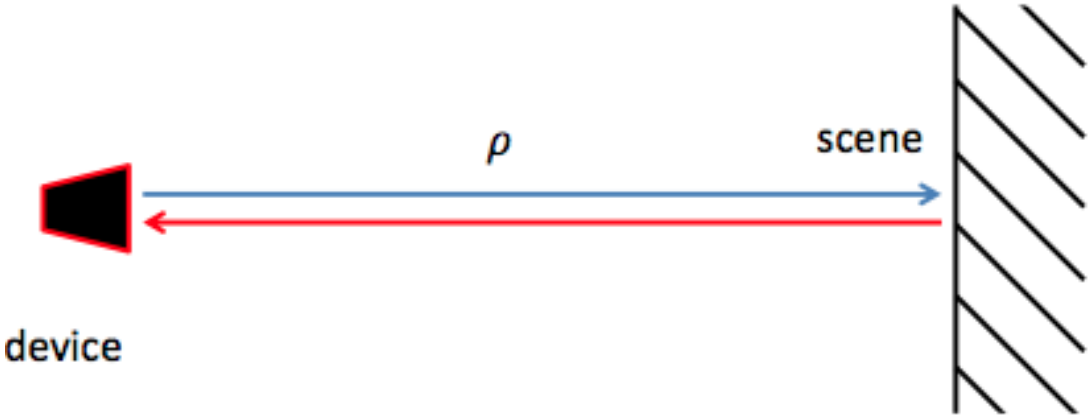


Figure 4.6 : measurment of TOF.

the radiation emitted at time 0 by the ToF sensor transmitter on the left travels straight towards the scene for a distance ρ, it is then reflected back by the scene surface, it travels back again for a distance ρ and at time τ it reaches the ToF sensor receiver, ideally co-positioned with the transmitter. Since at time τ the path length covered by the radiation is 2ρ, the relationship between ρ and τ in this case is :

$$\rho = \frac{c\tau}{2} \qquad (4.1)$$

which is the basis of ToF cameras distance measurements.

In spite of the conceptual simplicity of relationship , its implementation

hides tremendous technological challenges because it involves the light speed.

The implementation of this type of devices and their integration into matricial configurations are fundamental issues of current ToF systems development.

We can name PMD Vision Cam Cube as an example device of TOF camera, some of its features are:

- Near-infrared light (700-1400 nm)
- Continues wave modulation
- Sinusoidal signal
- Resolution: 204x204 pixels
- Standard lens, standard calibration
- Frame rate: 20 fps

• Multiple camera operation by using different modulation frequencies

Figure 4.7 : TOF camera.

4.4 Basics Of Imaging Systems And Kinect Operation

The Kinect is a proprietary product and its algorithms are still undisclosed, however some characteristics of the adopted light coding system can be deduced from its operation. The Kinect is a special case of 3D acquisition systems based on light coding. Understanding its operation requires a number of preliminary notions, such as the concepts of *pin-hole camera model*, *camera projection matrix* and *triangulation*.

4.4.1 Pin-hole camera model

Consider a 3D reference system (with axes *x*, *y* and *z*), called *Camera Coordinates System (CCS)*, with origin at *O*, called *center of projection*, and a plane parallel to the (*x*,*y*)-plane intersecting the z-axis at negative z-coordinate *f*, called *sensor* or *image plane S* as shown in Figure 4.8 The axis orientations follow the so-called right-hand convention. Consider also a 2D reference system

$$u = x + c_x \qquad (4.2)$$

$$v = y + c_y \qquad (4.3)$$

Associated to the sensor, called *S-2D reference system*, oriented as shown in Figure

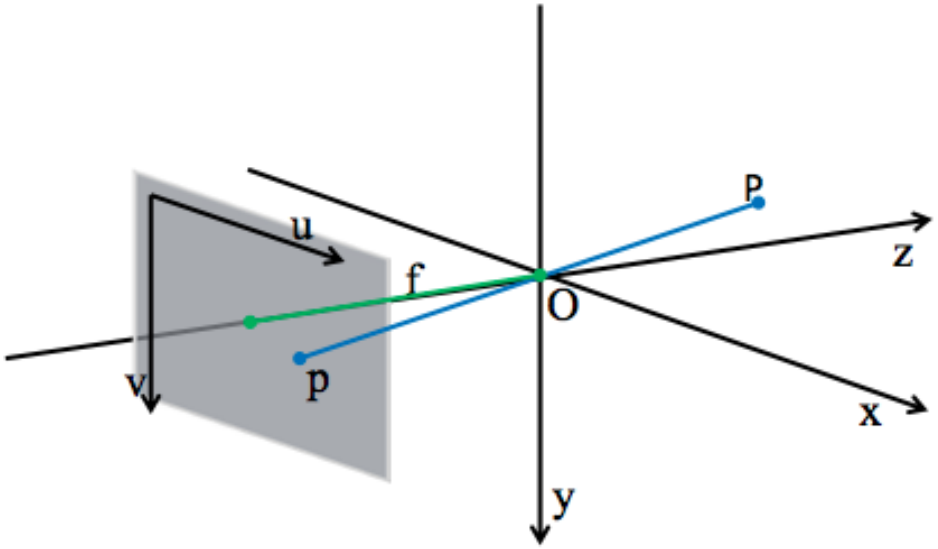
4.8 a. The intersection *c* of the z-axis with the sensor plane has coordinates c = [u = cx , v = cy]^T . The set of sensor points p, called *pixels*, of coordinates p = [u, v]^T obtained from the intersection of the rays connecting the center of projection *O* with all the 3D scene points p and P with coordinates P=[x,y,z]^T is the scene foot print on the sensor *S*. The relationship between *P* and *p*, called *central* or *perspective projection*.

Figure 4.8 a) shows the projections

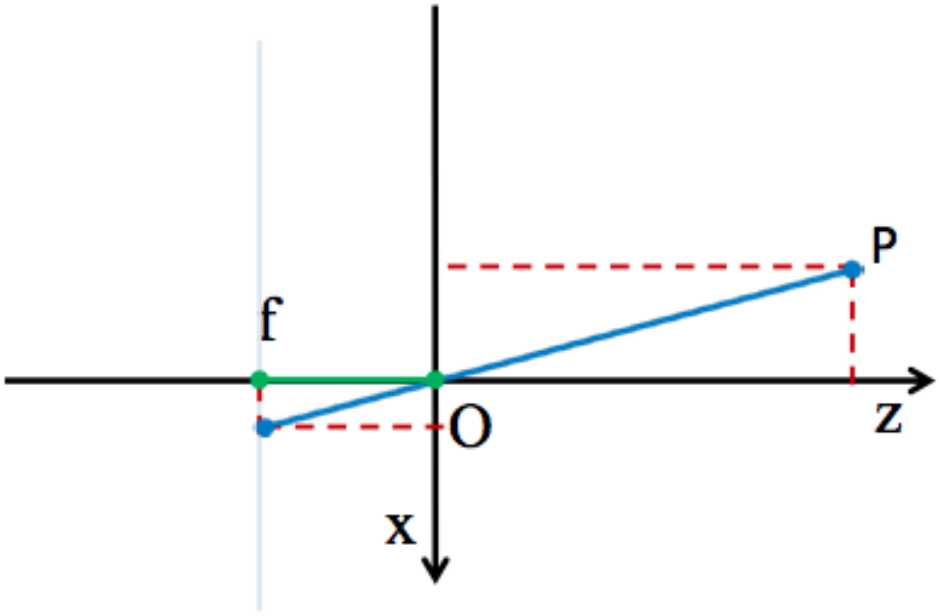
Perspective projection can be shown by triangles similarity(see Figure 4.8 b and c)[72]

So:

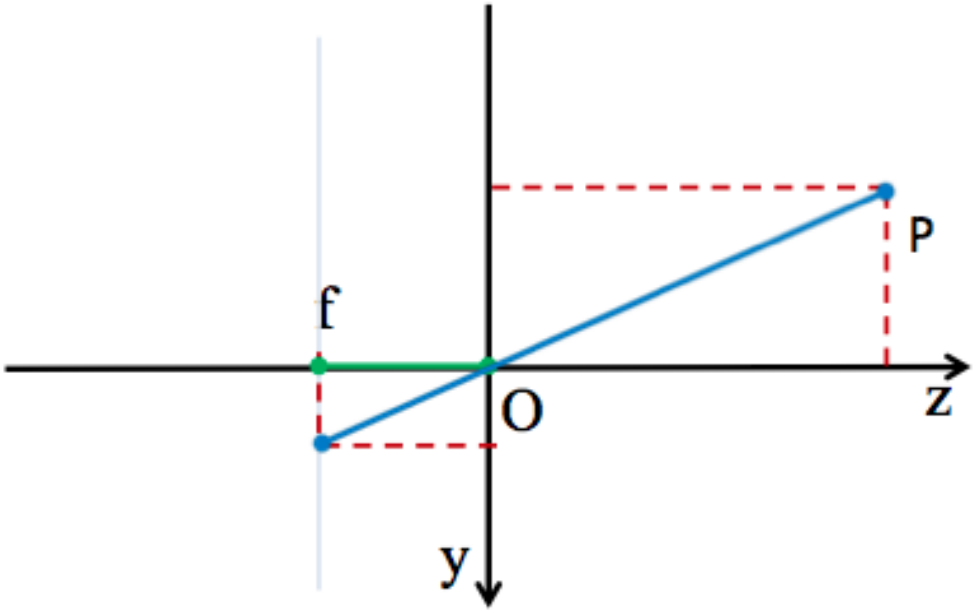
$$\begin{cases} u - c_x = f\dfrac{X}{Z} \\ v - c_y = f\dfrac{Y}{Z} \end{cases} \tag{4.4}$$



a)



b) $\frac{u-c_x}{f} = \frac{X}{z}$



c) $\frac{v-c_y}{f} = \frac{Y}{z}$

Figure 4.8 : Perspective projection: a)scene point P is projected to sensor pixel p; b)horizontal section of a); c) vertical section of a).

Perspective projection (4.4) is a good description of the geometrical relationship between the coordinates of the scene points and those of an image of them obtained by a pin-hole imaging device with pin-hole positioned at center of projection O. Such a system allows a single light ray to go throughout the pin-hole at O.

4.4.2 Intrinsic and extrinsic camera parameters

Projective geometry associates to each 2D point p with Cartesian coordinates p = [u, v]^T of a plane a 3D representation, called *homogeneous coordinates* $\tilde{p} = [hu, hv, h]^T$, where h is any real constant. The usage of h = 1 is rather common and $[u,v,1]^T$ is often called the *extended vector of p*.

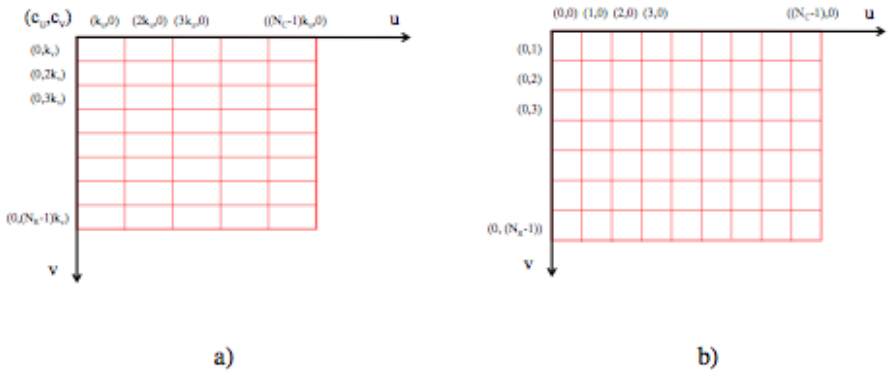
The coordinates of p=$[u,v]^T$ can be obtained from those of $\tilde{p} = [hu, hv, h]^T$ dividing them by the third coordinate h. Vector $\tilde{p}$ can be interpreted as the 3D ray connecting the sensor point p with the center of projection O.

In a similar way each 3D point P with Cartesian coordinates P = $[x,y,z]^T$ can be represented in homogeneous coordinates by a 4D vector $\tilde{p} = [hx, hy, hz, h]^T$ where h is any real constant. Vector $[x,y,z,1]^T$ is often called the *extended vector of P*. The coordinates of P = [x, y, z]^T can be obtained from P = [hx, hy, hz, h]^T dividing them by the fourth coordinate h.

The homogeneous coordinates representation of p allows to rewrite non-linear relationship (4.5) in a convenient matrix form, namely:

$$z\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad (4.5)$$

Note that the left side of (4.5) represents p in 2D homogeneous coordinates but the right side of (4.5) represents P in 3D Cartesian coordinates.Figure 4.9 shows the 2D 2D coordinates.



a) b)

Figure.4.9 : 2D sensor coordinates: a) rectangular window of a non-normalized orthogonal lattice; b) rectangular window of a normalized orthogonal lattice.

In order to deal with normalized lattices with origin at (0, 0) and unitary pixel coordinates $u_S \in [0, ..., N_C - 1]$ and $v_S \in [0, ..., NR - 1]$ in both u and v direction,

relationship (4.5) is replaced by

$$z\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K\begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad (4.6)$$

where *K* is the intrinsic parameters matrix defined as

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \qquad (4.7)$$

With $f_x = fk_u$ the *x*-axis focal length of the optics, $f_y = fk_v$ the *y*-axis focal length of the optics, c_x and c_y the (*u*,*v*) coordinates of the intersection of the optical axis with the sensor plane. All these quantities are expressed in [*pixel*], i.e., since *f* is in [*mm*], k_u and k_v are assumed to be [*pixel*]/[*mm*].

4.4.3 Stereo vision systems

A stereo vision (or just *stereo*) system is made by two standard (typically identical) cameras partially framing the same scene. Such a system can always be calibrated and rectified [72]. It then becomes equivalent to a stereo vision system made by two (identical) standard cameras with coplanar and aligned imaging sensors and parallel optical axis.

Let us introduce next the stereo vision notation. The two cameras of the (calibrated and rectified) stereo vision system *S* are the left camera *L* (also called *reference camera*) and the right camera *R* (also called *target camera*). Each camera, as seen above, has its own 3D CCS and 2D reference systems as shown in Figure 4.10 Namely the *L* camera has CCS with coordinates (x_L, y_L, z_L), also called L-3D *reference system*, and a 2D reference system with coordinates (u_L, v_L). The *R* camera has CCS with coordinates (x_R, y_R, z_R), also called *R-3D reference system*, and a 2D reference system with coordinates (u_R,v_R). A common convention is to consider the *L*-3D reference system as the reference system of the stereo vision system, and to denote it as *S*-3D reference system.

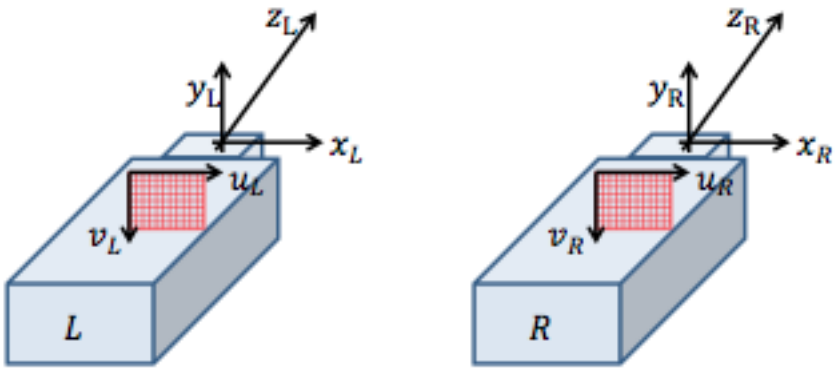


Figure 4.10 : Stereo vision system coordinates and reference systems.

In the case of a calibrated and rectified stereo vision system, a 3D point *P* with coordinates P=[*x*,*y*,*z*]^T with respect to the *S*-3D reference system is projected to the pixels P_L and P_R of the *L* and *R* cameras with coordinates $P_L = [u_L, v_L]^T$ and $P_R = [u_R = u_L - d, v_R = v_L]^T$ respectively as shown in Figure 4.10. From the relationships concerning the triangle with vertices at P_R, *P* and P_L it can be shown that in a rectified setup, where points P_L and P_R, have the same vertical coordinates, the difference between their horizontal coordinates $d = u_L - u_R$, called *disparity*, is inversely proportional to the depth value *z* of *P* through the well-known triangulation relationship

$$Z = \frac{bf}{d} \qquad (4.8)$$

which In *b* is the *baseline*, i.e, the distance between the origins of the *L*-3D and the *R*-3D reference systems, and f is the focal length of both cameras.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = K_L^{-1} \begin{bmatrix} u_L \\ v_L \\ 1 \end{bmatrix} z \qquad (4.9)$$

Where K_L^{-1} is the inverse of the rectified intrinsic parameters matrix of camera L. So the 3D coordinates P=[*x*,*y*,*z*]^T of a scene point *P* can be computed from (4.8) and (4.9), usually called triangulation or computational stereopsis.

Detecting conjugate pixels between the stereo image pair, typically called the correspondence problem, is one of the major challenges of a stereo vision algorithm. The methods proposed for this task are typically divided in *local* and *global* approaches. Local methods consider only local similarity measures between the region surrounding p_L and regions of similar shape around all the candidate

conjugate points p_R of the same row. The selected conjugate point is the one maximizing the similarity measure, a method typically called Winner Takes All strategy. Global methods do not consider each couple of points on its own but estimate all the disparity values at once exploiting global optimization schemes. Global methods based on Bayesian formulations are currently receiving great attention. Such techniques generally model the scene as a Markov random field and include within a unique framework cues coming from the local comparisons between the two images and scene depth smoothness constraints. Global stereo vision algorithms typically estimate the disparity image by minimizing a cost function made by a *data term* representing the cost of local matches, similar to the one of local algorithms (e.g., covariance) and a *smoothness term* defining the smoothness level of the disparity image by explicitly or implicitly accounting for discontinuities [72].

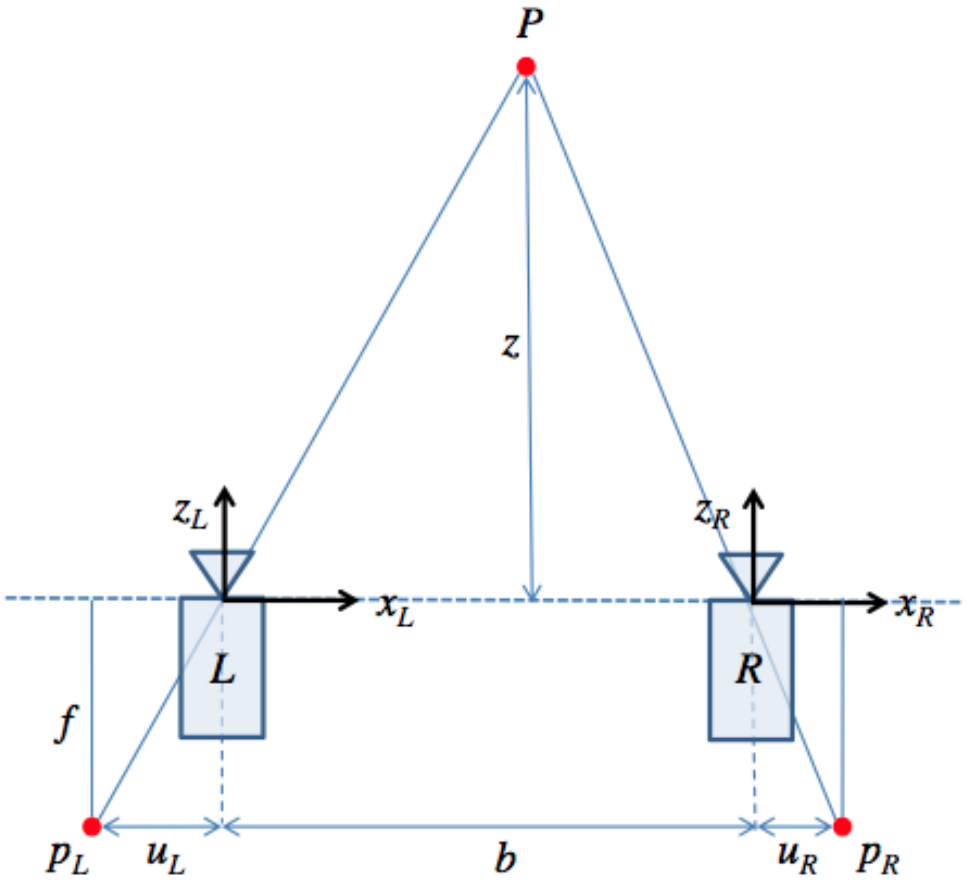


Figure 4.11 : Triangulation with a pair of aligned and rectified cameras.

A stereo vision system where one of the two cameras is replaced by a projector, made by a camera *C* and a projector *A* as shown in Figure 4.12, is called *active* or

light coding system. Camera C has CCS system with coordinates (x_c, y_c, z_c) also called *C-3D reference system* and a 2D reference system with coordinates (u_c, v_c) as shown in Figure 4.12 Projector *A* similarly has CCS with coordinates (x_A, y_A, z_A) also called *A-3D reference system* and a 2D reference system with coordinates (u_A, v_A).

As in the case of standard passive stereo systems, active systems of the type shown in Figure 4.12 can be calibrated and rectified in order to simplify the depth estimation process. [72]

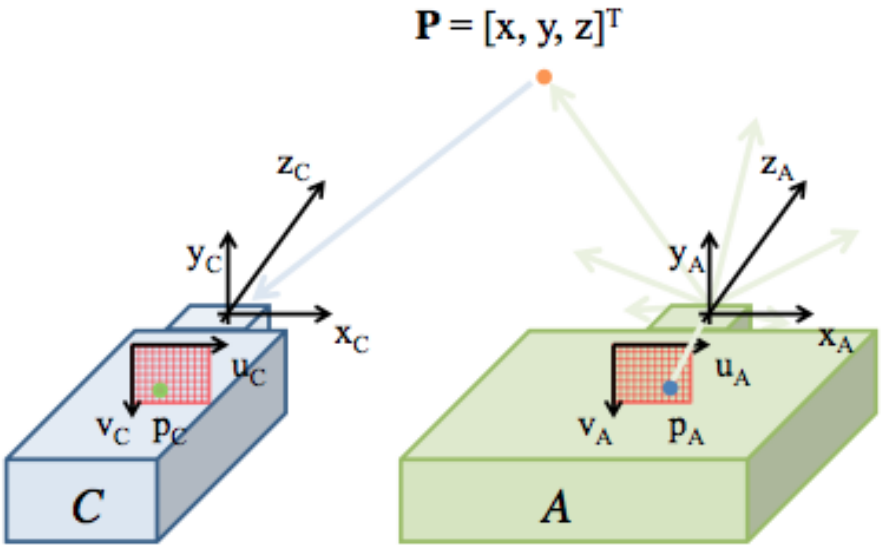


Figure 4.12 : Active triangulation by a system made by a camera C (blue) and a light projector A (green).

Figure 4.12 shows the projection of light pattern pixel P_A with coordinates $p_A = [u_A, v_A]^T$ in the *A*-2D reference system to 3D scene point *P* with coordinates $P = [x, y, z]^T$ in the *C*-3D reference system. If *P* is not occluded it projects the light radiant power received by the projector to pixel P_c of camera C establishing triangle $p_c P p_A$.

Namely, given a point P_c in the image acquired by C and its conjugate point P_A in the pattern projected by *A*, the depth Z of the 3D point P associated to P_c with respect to the *C*-3D reference system is computed by active triangulation as described below. In order to use a pattern invisible to human eyes, both projector and camera operate at IR wavelengths. The Kinect applies active triangulation to all the $NR \times N_C$ pixels of I_K , a procedure called matricial *active triangulation*.

The Kinect range camera has the structure shown in Figure 4.12 reported also in Figure 4.13 [72] with a camera *C* and a projector *A*, and in principle it implements active triangulation.

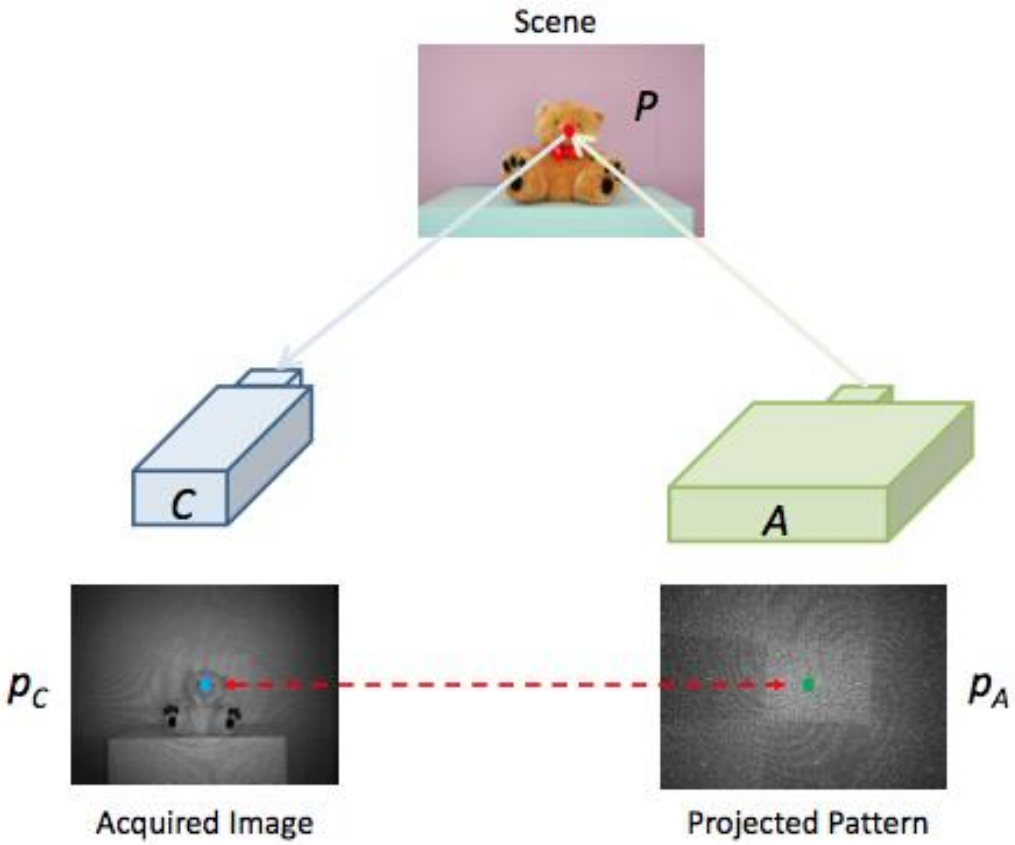


Figure 4.13 : Matricial active triangulation flow: pixel pA (green dot) is coded in the pattern. The pattern is projected to the scene and acquired by C. The 3D point associated to pA is P and the conjugate point of pA (green dot) in IK is pC (blue dot). The correspondence estimation algorithm (red dashed arrow) estimates the conjugate points.

4.5 Depth Sensors In TOF Camera

The ToF depth sensor emits IR waves to target objects, and measures the phase delay of reflected IR waves at each sensor pixel, to calculate the distance travelled. According to the color, reflectivity and geometric structure of the target object, the reflected IR light shows amplitude and phase variations, causing depth errors. Moreover, the amount of IR is limited by the power consumption of the device, and therefore the reflected IR suffers from low signal-to-noise ratio (SNR). To increase the SNR, ToF sensors bind multiple sensor pixels to calculate a single depth pixel value, which decreases the effective image size. Structured light depth sensors

project an IR pattern onto target objects, which provides a unique illumination code for each surface point observed at by a calibrated IR imaging sensor. Once the correspondence between IR projector and IR sensor is identified by stereo matching methods, the 3D position of each surface point can be calculated by triangulation.In both sensor types, reflected IR is not a reliable cue for all surface materials. For example, specular materials cause mirror reflection, while translucent materials cause IR refraction. Global illumination also interferes with the IR sensing mechanism, because multiple reflections cannot be handled by either sensor type.

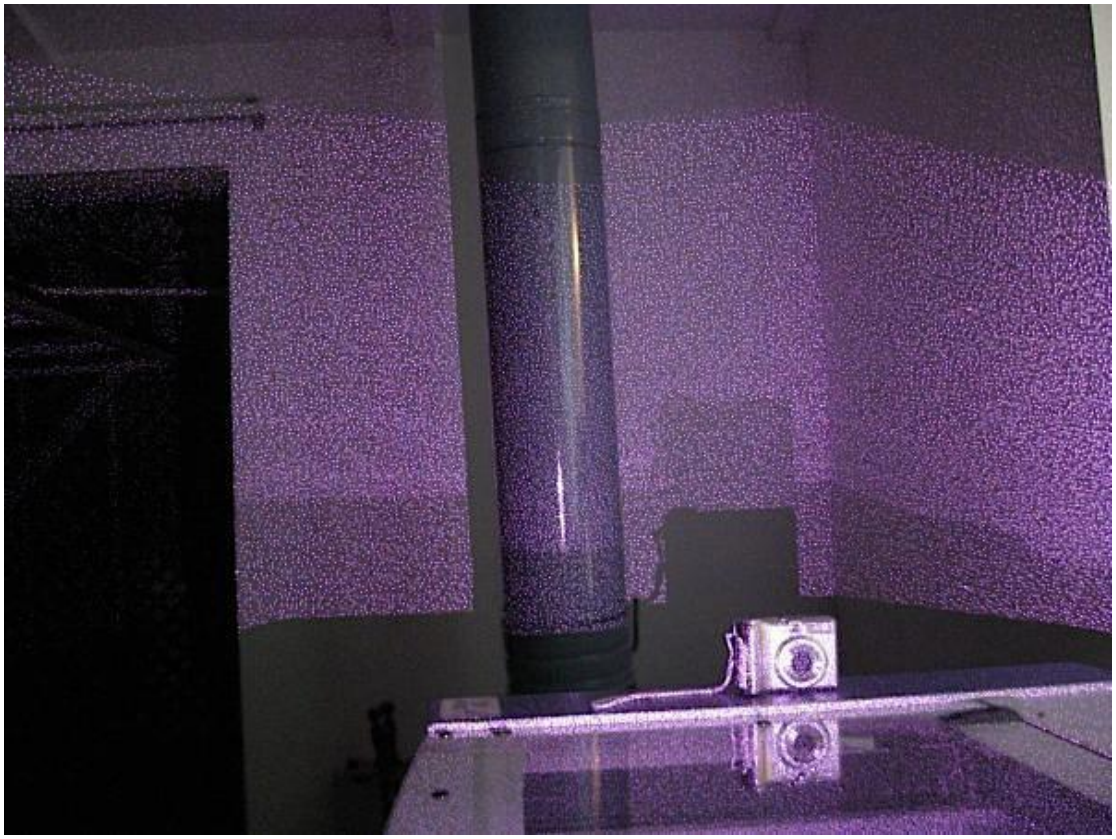


Figure 4.14: IR reflection.

4.5.1 Standard depth data set

A range of commercial ToF depth cameras have been launched in the market, such as PMD, PrimeSense, Fotonic, ZCam, SwissRanger, 3D MLI, and others. Kinect is the first widely successful commercial product to adopt the IR structured light principle. Among many possibilities, we specifically investigate two depth cameras: a ToF type SR4000 from MESA Imaging , and a structured light type Microsoft Kinect are the most popular depth cameras in the research community, accessible in the market and reliable in performance.

Mesa SR4000. This is a ToF type depth sensor producing a depth map and amplitude image at the resolution of 176 × 144 with 16 bit floating-point precision. The amplitude image contains the reflected IR light corresponding to the depth map. In addition to the depth map, it provides $\{x, y, z\}$ coordinates, which correspond to each pixel in the depth map.

Kinect. This is a structured IR light type depth sensor, composed of an IR emitter, IR sensor and color sensor, providing the IR amplitude image, the depth map and the color image at the resolution of 640 × 480 (maximum resolution for amplitude and depth image) or 1600 × 1200 (maximum resolution for RGB image).

Time-of-flight cameras can, in principle, be modeled and calibrated as pinhole devices. For example, if a known chequer board pattern is detected in a sufficient variety of poses, then the internal and external camera parameters can be estimated by standard routines .

Figure 4.15 : XBOX 360.

4.6 The Kinect 2.0

Details about the next version of Microsoft's Kinect, to be bundled with the upcoming Xbox One, are slowly emerging. After an initial leak of preliminary specifications on February 20th, 2013, finally some official data are to be had. This article about the upcoming next Kinect-for-Windows mentions "Microsoft's proprietary Time-of-Flight technology," which is an entirely different method to

sense depth than the current Kinect's structured light approach. That's kind of a big deal.

Given that additional bit of information, the leaked depth camera specs make a lot more sense. According to the leak, the new Kinect ("Kinect2″ from here on out) has a depth camera resolution of 512×424 pixels. It was surprising initially, given that Kinect1′s depth camera has a resolution of 640×480 pixels. But, the Xbox 360 only used a depth image of 320×240 pixels for its skeletal tracking, mostly for performance reasons. So at first guessed that the new Xbox would again only use a down sampled depth image, and that the leaked resolution was the down sampled one, leading to a "true" depth resolution of 1024×848 pixels.

But there was a problem: the Kinect1′s depth camera is not a real camera; it's a virtual camera, created by combining images from the real IR camera (which has 1280×1024 resolution) with light patterns projected by the IR emitter. And therein lies the rub. While the virtual depth camera's nominal resolution is 640×480, the IR camera can only calculate a depth value for one of its (real) pixels if that pixel happens to see one of the myriad of light dots projected by the pattern emitter. And because the light dots must have some space between them, to be told apart by the IR camera, and to create a 2D pattern with a long repetition length, only a small fraction of the IR camera's pixels will see light dots in any given setting. The depth values from those pixels will then be resampled into the 640×480 output image, and depth values for all other pixels will be created out of thin air, by interpolation between neighboring real depth values.

The bottom line is that in Kinect1, the depth camera's nominal resolution is a poor indicator of its effective resolution. Roughly estimating, only around 1 in every 20 pixels has a real depth measurement in typical situations. This is the reason Kinect1 has trouble detecting small objects, such as fingertips pointing directly at the camera. There's a good chance a small object will fall entirely between light dots, and therefore not contribute anything to the final depth image. This also means that simply increasing the depth camera's resolution, say to 1024×848, without making the projected IR pattern finer and denser as well, would not result in more data, only in more interpolation. But in new type the technology is changed!

In a time-of-flight depth camera, the depth camera is a real camera (with a single real

lens), with every pixel containing a real depth measurement. This means that, while the nominal resolution of Kinect2′s depth camera is lower than Kinect1′s, its effective resolution is likely much higher, potentially by a factor of ten or so. Time-of-flight depth cameras have their own set of issues, so it is expecting much more detailed depth images. From a purely technical point of view, Kinect2 does use a time-of-flight depth camera, and if that camera's native resolution really is 512×424, that's a major achievement in itself. As of now, time-of-flight cameras have very low resolutions, usually 160×120 or 320×240. Even Intel/Creative's upcoming depth camera is reported to use 320×240, or a factor of three fewer pixels than Kinect2.

Structured-light depth cameras have another subtle drawback. To measure the depth of a point on a surface, that point has to be visible to both the camera and pattern emitter. This leads to distinct "halos" around foreground objects. More distant surfaces on the left side of the foreground object can't be seen by the camera, whereas surfaces on the right side can't be seen by the pattern emitter (or the other way around, depending on camera layout). The larger the depth distance between foreground and background objects, the wider the halo. A time-of-flight camera, on the other hand, can measure the depth of any surfaces it can see itself. In truth, there is still an emitter involved; the emitter needs to create a well-timed pulse of light whose return time can be measured. But since depth resolution does not linearly depend on the distance between the camera and emitter, the emitter can be very close to the camera — it can even shoot through the same lens — and the resulting halos are much smaller, or gone completely.

So is the higher depth resolution just an incremental improvement, or a major new feature? For some applications, like skeleton tracking or 3D video, it is indeed only incremental, albeit highly welcome. But there are very important applications for which Kinect1′s depth resolution was *barely* not good enough, most importantly finger and face tracking. Based on the known specs, It is expecting that Kinect2′s depth camera will be able to resolve finger tips at medium distance reliably, even when pointing directly at the camera. This will enable new natural user interfaces for 3D interactions, such as grabbing, moving, rotating, and scaling virtual three-dimensional objects. Reliable face tracking could be used to create truly holographic 3D displays. These significant improvements in the depth camera aside, the other changes are really quite minor. Kinect2 has a higher-resolution color camera, which

can allegedly stream 1920×1080 pixel color images at 30 Hz, compared to Kinect1′s 640×480 pixel images at 30 Hz, or 1280×1024 pixel images at 15 Hz. Because it was already possible to combine the Kinect with external cameras, that’s not that important. And the new microphone array seems to be basically the same as the old.

4.7 Conclusion And Further Reading

Even though kinect originally introduced for gesture recognition in the gaming context [74], the Kinect has readily been adopted for several other purposes such in robotics [75] and 3D scene reconstruction [76]. Already just one year after its introduction, the number of applications based on Kinect is dramatically growing and its usefulness and relevance for 3D measurement purposes are becoming apparent.

The Kinect range camera is based on a light-coding technique and neither Microsoft nor other brand have disclosed yet all the sensor implementation details. Several patents, among which , cover the technological basis of the range camera, and the interested reader might look at them or to current reverse engineering works [77, 78].

This chapter presents the Kinect range camera emphasizing its general operation principles rather than its implementation details, not only because the latters are not available, but also because the technology behind Kinect is likely to rapidly evolve. In a longer term perspective the general light coding operation principles are likely to be more useful than the characteristics of the first generation product. A comprehensive review of light-coding techniques can be found in [79].

5. IMAGE BASED VISUAL SERVO CONTROL BASED ON DEPTH CAMERA

Depth camera based visual servoing has many advantages over the classical monocular visual servoing and stereovision approaches .The depth information can be recovered without any geometrical model of the observed object. It should be noted that even in image based visual servoing, this information is needed for the computation of the image interaction matrix. In this chapter modeling of the robotic, the image processing and feature extraction procedures are discussed, depth camera based visual servoing system and also simulation results for developed methods and algorithms are presented.

As it was explained in previous chapters there is two major approaches in visual servo control, Image based visual servo control and position based visual servo control according to its advantage and stability that was explained in preceding chapters, we choose IBVS method. This research work aims at introducing a depth camera based open loop and close loop, eye- in- hand image based visual servoing system and eye to hand image based visual servoing. In this research we have addressed the problem of grasping as an eye-in-hand visual servoing problem, which can be expressed and defined as “finding the (proper) motion of the manipulator that will grasp a moving object with limited motion velocity”. With regards to the visual servoing problem the procedure of grasping can be also stated as: “finding the motion of the manipulator that will cause the image projections of target’s feature points to move and coincide to the desired image features”. This can be done by pre-defining desired positions for the object’s image features such that the robot moves and aligns the end-effector with the object and reaches towards it. By using a depth camera such as kinect we can extract the exact depth and 3-D position information that helps the procedure of servoing and grasping to be more accurate.

5.1 Camera Parameters

5.1.1 Perspective projection equation:

To get a clear understanding about the intrinsic and extrinsic parameters of a camera, the concept of perspective projection equation must be considered. According to the perspective projection model a point ${}^{C}P = [x, y, z]^{T}$, whose coordinates are expressed with respect to the camera coordinate frame, will project onto the image plane with the coordinates $p = [u, v]^{T}$, given by

$$P(x, y, z) = \begin{bmatrix} u \\ v \end{bmatrix} = \frac{\lambda}{z} \begin{bmatrix} x \\ y \end{bmatrix} \qquad (5.1)$$

Where λ is the focal length of the lens. The terms x, y, z, u, v and λ are shown in the following Figure (5.1).

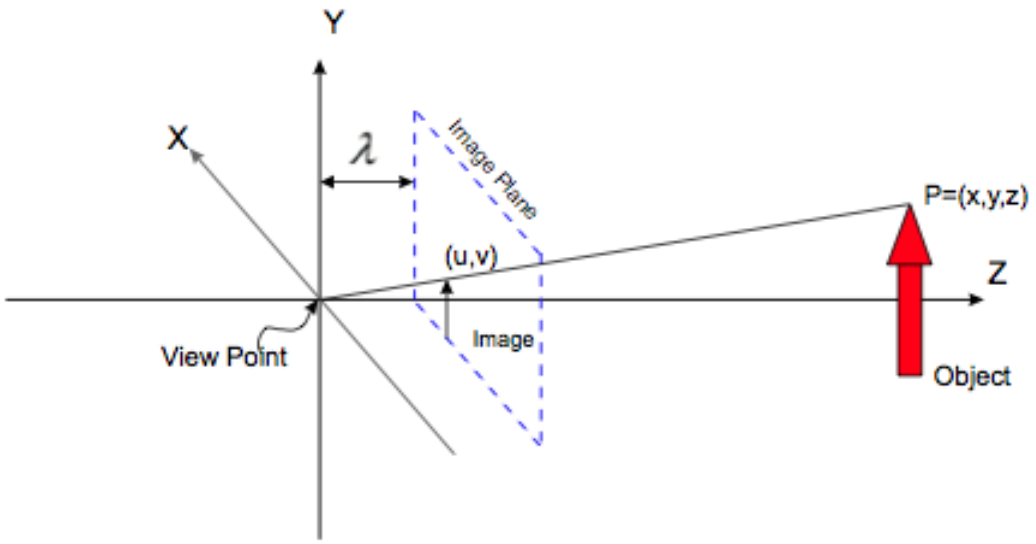


Figure 5.1 : Coordinate frame for the camera/lens system.

The coordinates (x, y, z) of a scene point P observed by a pinhole camera are related to its image coordinates (u, v) by the perspective equation. In practice, the world, camera and image coordinate systems are related by a set of physical parameters such as the focal length of the lens, size of the pixels, the position of the principal point, and the position and orientation of the camera, translation and rotational changes. These can be classified into two categories. They are extrinsic parameters and intrinsic parameters.

5.1.2 Extrinsic parameters

The camera frame (C) and the world frame (W) are different. So there should be a transformation from the world coordinate frame to the camera frame.

$$C_p = ({}^{w}_{c}R \; {}^{w}_{c}o \,) \begin{pmatrix} W_p \\ 1 \end{pmatrix} \qquad (5.2)$$

Where R = ${}^{C}_{W}$ R is a rotation matrix and t = ${}^{w}_{c}o$ is a translation vector and P denotes the vector of homogeneous coordinates of P in the frame (W).

5.1.3 Intrinsic parameters

As it was mentioned in preceding chapter as kinect somehow looks like to pinhole camera so we can talk about its intrinsic parameters like pinhole. The camera has two different image planes: the first one is a normalized plane located at a unit distance from the pinhole. This plane has its own coordinate system with an origin located at the point $\hat{C}$ where the optical axis pierces it. The perspective projection equation in the normalized form can be written as

$$\begin{cases} \hat{u} = \dfrac{x}{z} \\ \hat{v} = \dfrac{y}{z} \end{cases} \qquad (5.3)$$

Where $\hat{p} = (\hat{u}\,, \hat{v}\,, 1)^T$ is the vector of homogeneous coordinates of the projection $\hat{p}$ of the point P into the normalized image plane. The physical retina of the camera is in general different. It is located at a distance $\lambda \neq 1$ from the pinhole, and the image coordinates (u,v) of the image point p are usually expressed in pixel units. In addition, pixels are normally rectangular instead of square, so the camera has two additional scale parameters k and l;

$$u = k.\lambda \, \frac{x}{z} \qquad (5.4)$$

$$v = l.\lambda \, \frac{y}{z} \qquad (5.5)$$

The parameters k, l and λ are not independent, and they can be replaced by the magnifications $\alpha = k\lambda$ and $\beta = l\lambda$ expressed in pixel units. Now, in general, the actual origin of the camera coordinate system is at a corner C of the retina (e.g., in the case depicted in figure above, the lower-left corner, or sometimes the upper-left

corner, when the image coordinates are the row and column indices of a pixel) and not at its center, and the center of the physical retina plane usually does not coincide with the principal point C_O. This adds two parameters u_O and v_O that define the position (in pixel units) of C_O in the retinal coordinate system.

Figure 5.2 shows the Physical and normalized coordinate systems [79].

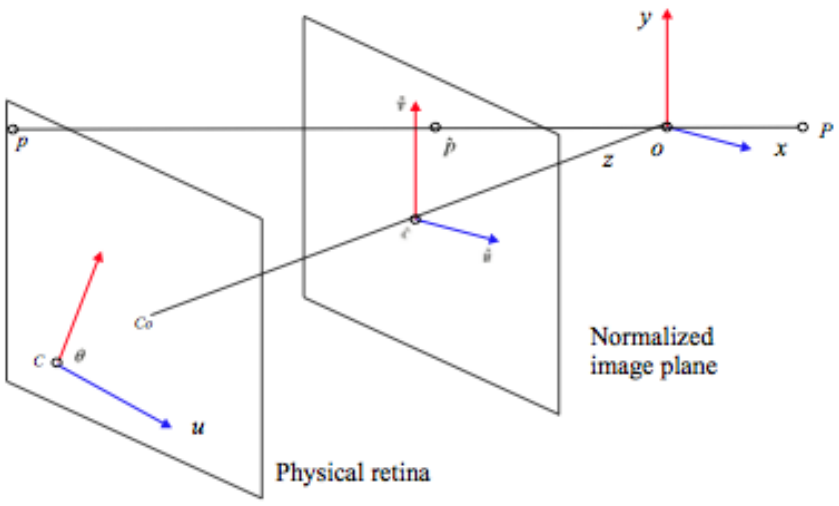


Figure 5.2 : Physical and normalized coordinate systems.

Thus eq.(5.4) ,(5.5) is replaced by

$$u = \alpha \frac{x}{z} + u_0 \qquad (5.6)$$

$$v = \beta \frac{y}{z} + v_0 \qquad (5.7)$$

5.2 Image Processing And Feature Extraction

Using vision sensor is particularly interesting since it is rich in information that the camera could provide and the variety of tasks that could be performed. This richness requires particular image processing techniques in order to extract the information that could be used in control. These techniques should provide real-time performance. In this project some already existing image processing techniques are going to be used. Since our interest is the control part, the focus of the literature review will be for visual servoing where we will consider that feature extraction is available. However, the accuracy of feature extraction would be taken into account to

assess visual servoing methods. First, features or measures should be extracted from images that will also represent the error space, after that we should produce the control commands from the error. Having selected IBVS approach, we need to choose some good features that are easy to detect and track in a sequence of frames accurately, as the main development of IBVS is done for point features, we choose edges points of the shape. In order to come up with a reliable image processing and feature extraction algorithm, the first thing to decide is which color space to found the detection algorithms on. We implemented our methods using two different color spaces: RGB and HSV.

5.2.1 RGB-based feature extraction

The RGB color model is an additive color model in which red, green, and blue light are added together in various ways to reproduce a broad array of colors (Figure 5.3) [80].

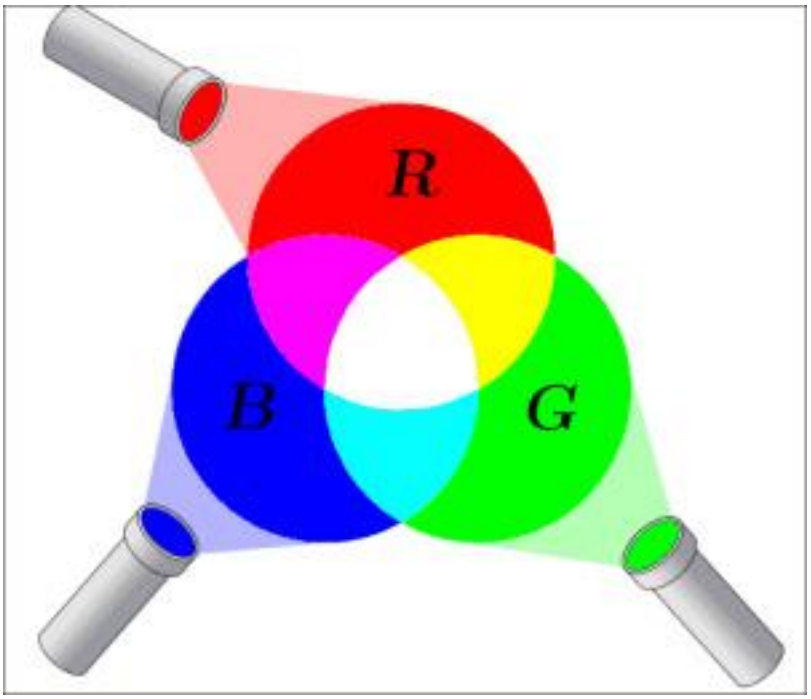


Figure 5.3 : RGB color space.

The name of the model comes from the initials of the three additive primary colors, red, green, and blue . The main purpose of using the RGB color model is for the sensing, representation, and display of images in electronic systems, such as televisions and computers, though it has also been used in conventional photography. RGB is a convenient color model for computer graphics because the human visual

system works in a way that is similar to an RGB color space.

Image object detections based on color is one of the quickest and easiest methods for tracking an object from one image frame to the next. The speed of this technique makes it very popular for real-time applications.

Since changing the brightness and light in workspace causes a lot of differences in the captured colors, we considered tolerances for Red, Green and Blue values to detect the objects based on RGB color space. Predefined RGB values for each feature color are saved then for each frame captured by the vision system all of the pixels in the image are examined and the pixels detected to be in the range of the saved RGB values with considered tolerances are marked. According to Equations (5.1-5.2) and having the total number of detected pixels (N) and also the position of each in the Cartesian frame of the image, it is possible to calculate the centroid of the detected object $(x_{centroid}, y_{centroid})$:

$$x_{centroid} = \frac{1}{N}\sum_{n=0}^{N} x_n \qquad (5.8)$$

$$y_{centroid} = \frac{1}{N}\sum_{n=0}^{N} y_n \qquad (5.9)$$

5.2.2 HSV-based feature extraction

HSV is one of the most common cylindrical-coordinate representations of points in an RGB color model, which rearrange the geometry of RGB in an attempt to be more perceptually relevant than the Cartesian representation [69] .

They were developed in the 1970s for computer graphics applications, and are used for color pickers, in color- modification tools in image editing software, and less commonly for image analysis and computer vision. HSV stands for hue, saturation, and value, and is also often called HSB (B for brightness). In each cylinder, the angle around the central vertical axis corresponds to "hue", the distance from the axis corresponds to "saturation", and the distance along the axis corresponds to "lightness", "value" or "brightness". The most important flaw about RGB-Based detection methods is that changing the light and brightness of the framework and work-space affects detection so badly that sometimes the program cannot identify the color objects at all. Knowing the fact that the brightness is an independence value in HSV (or HSB) color space, we can come up with a proper tolerance to be considered

in program, which covers all the values of a predefined color in the workspace. This will certainly make the implementation of the color detection more stable.

Figure 5.4 [80] shows an illustration of HSV color space.

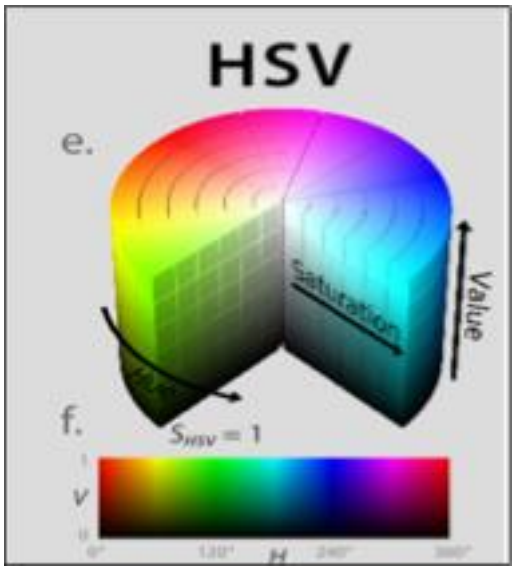


Figure 5.4 : HSV color space.

5.3 Camera-Robot System

The pose of an object is defined as its position and orientation. The position in 3D Euclidean space is given by the 3 Cartesian coordinates. The orientation is usually expressed by 3 angles, i.e. the rotation around the 3 coordinate axes. Figure 5.5 shows the notation used in this chapter, where yaw, pitch and roll angles are defined as the mathematically positive rotation around the *x*, *y* and *z* axis.

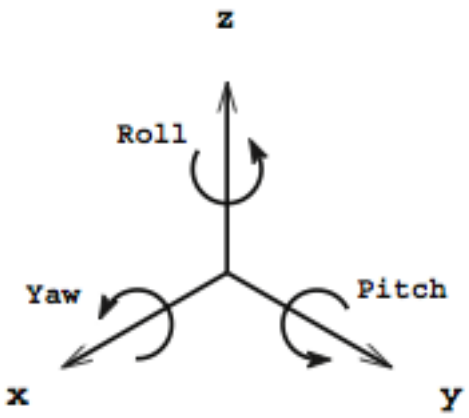


Figure 5.5 : Yaw, pitch and roll.

In this chapter we use the {·}-notation for a coordinate system, for example {*W*}

will stand for the world coordinate system. A variable coordinate system—one which changes its pose to over time—will sometimes be indexed by the time index $n \in$ IN = 0, 1, 2, ... An example is the camera coordinate system $\{Cn\}$, which moves relative to $\{W\}$ as the robot moves since the camera is mounted to its hand. Figure 5.6 lists the coordinate systems used for modeling the camera-robot system.

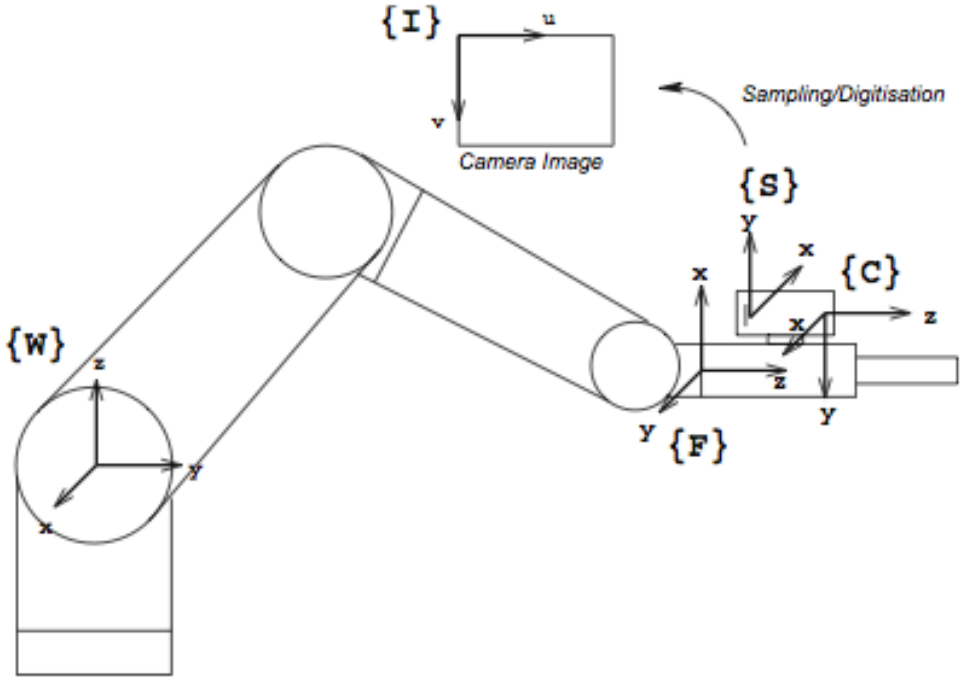


Figure 5.6 : World, Flange, Camera, Sensor and Image coordinate systems.

The world coordinate system $\{W\}$ is fixed at the robot base, the flange coordinate system $\{F\}$ (sometimes called "tool coordinate system", but this can be ambiguous) at the flange where the hand is mounted. The camera coordinate system $\{C\}$ (or $\{Cn\}$ at a specific time n) is located at the optical center of the camera, the sensor coordinate system $\{S\}$ in the corner of its CCD/CMOS chip (sensor); their orientation and placement is shown in the figure. The image coordinate system which is used to describe positions in the digital image is called $\{I\}$. It is the only system to use pixel as its unit; all other systems use the same length unit.

5.3.1 Defining the camera-robot system as a dynamical system

As mentioned before, the camera-robot system can be regarded as a dynamical system. We define the state x_n of the robot system at a time step $n \in N$ as the current robot pose, i.e. the pose of the flange coordinate system $\{F\}$ in world coordinates $\{W\}$. $x_n \in IR^6$ will contain the position and orientation in the x, y, z, yaw, pitch, roll

notation defined above. The set of possible robot poses is X$\subset IR^6$. The output of the system is the image feature vector y_n. It contains pairs of image coordinates of object markings viewed by the camera, i.e.$(x_1, y_1, \dots x_M, y_M)^T$ for M=$\frac{m}{2}$ object markings (in our case *M=4*, so $y_n \in IR^8$). Let Y$\subset IR^m$ be the set of possible output values. The output (measurement) function is, $\eta : X \to Y, x_n \mapsto y_n$

It contains the whole measurement process, including projection on to the sensor, digitization and image processing steps.

The input (control) variable $u_n \in U \subset IR^6$ shall contain the desired pose change of the camera coordinate system. This robot movement can be easily transformed to a new robot pose $\widetilde{u_n}$ in $\{W\}$, which is given to the robot in a move command. Using this definition of u_n an input of $(0,0,0,0,0,0)^T$ corresponds to no robot movement, which has advantages, as we shall see later. Let $\varphi : X \times U \to X, (x_n, u_n) \mapsto x_{n+1}$ be the corresponding state transition (next-state) function.

With these definitions the camera-robot system can be defined as a time invariant, time discrete input-output system:

$$x_{n+1} = \varphi(x_n, u_n) \qquad (5.10)$$

$$y_n = \eta\,(x_n) \qquad (5.11)$$

For *m* = 2 image features corresponding to coordinates (^{S}x, ^{S}y) of a projected object point ^{W}p the equation for η follows analogously:

$$\eta(x) = y = {}^{s}y \qquad (5.12)$$

5.3.2 The forward model—mapping robot movements to image changes

In order to calculate necessary movements for a given desired change in visual appearance the relation between a robot movement and the resulting change in the image needs to be modeled. In this section we will analytically derive a forward model, i.e. one that expresses image changes as a function of robot movements, for the eye-in-hand setup described above. This forward model can then be used to predict changes effected by controller outputs, or (as it is usually done) simplified and then inverted. An inverse model can be directly used to determine the controller

output given actual image measurements.

Let $\phi : X \times U \rightarrow Y$ the function that expresses the system output *y* depending on the state *x* and the input *u*:

$$\phi(x,u) := \eta \; o \; \varphi(x,u) = \; \eta \; (\varphi(x,u)) \qquad (5.13)$$

For simplicity we also define the function which expresses the behavior of $\Phi(x_n, \cdot)$ at a time index *n*, i.e. the dependence of image features on the camera movement *u*:

$$\phi_n(u) := \phi(x_n,u) = \eta \; (\varphi(x_n,u)) \qquad (5.14)$$

5.4 Using Image Features

In this thesis for extracting the features we considered point features method. The points would be the centroids of distinct regions, or Harris or SURF corner features. The points would then be used for pose estimation in a PBVS scheme, or directly in an IBVS scheme. (we use IBVS scheme)For both PBVS or IBVS we need to solve the correspondence problem, that is, for each observed feature we must determine which desired image plane coordinate it corresponds to. IBVS can also be formulated to work with other image features such as lines, as found by the Hough transform, or the shape of an ellipse.

5.5 Designing a Visual Servoing Controller

In a digital camera the image plane is a $W \times H$ grid of light sensitive elements called photo sites that correspond directly to the picture elements (or pixels) of the image as shown in Fig.5.7 The pixel coordinates are a 2-vector (u, v) of non-negative integers and by convention the origin is at the top-left hand corner of the image plane.

The pixels are uniform in size and centered on a regular grid so the pixel coordinate is related to the image plane coordinate by

$$u = \frac{x}{\rho_w} + \; u_0 \qquad (5.15)$$

$$v = \frac{y}{\rho_h} + \; v_0 \qquad (5.16)$$

Where ρ_w and ρ_h are the width and height of each pixel respectively, and (u_0, v_0) is

the principal point, the coordinate of the point where the optical axis intersects the image plane.

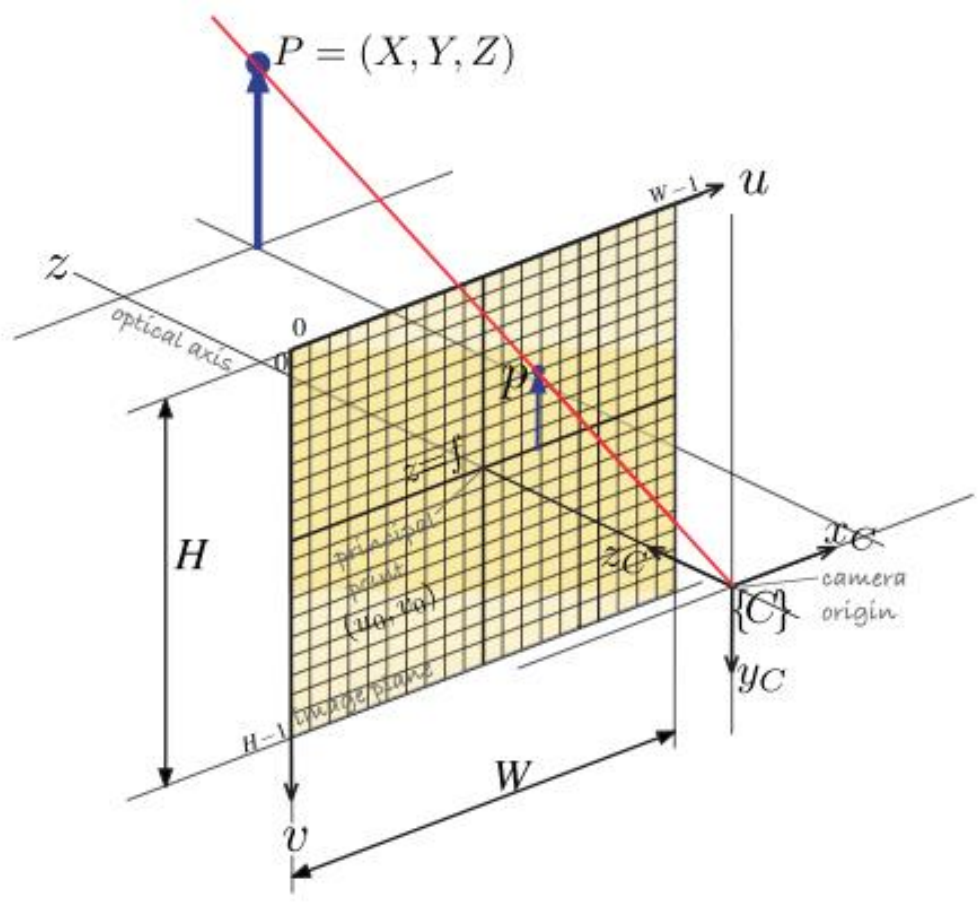


Figure 5.7 : Central projection model showing image plane and discrete pixels.

The projection can also be written in functional form as

$$p = \mathcal{P}(P, K, \xi_c) \qquad (5.17)$$

Where P is the point in the world frame, ***K*** is the camera parameter matrix and ξ_c is the pose of the camera with respect to the world coordinate frame.

$$\hat{p} = \begin{pmatrix} \frac{1}{\rho_w} & 0 & u_0 \\ 0 & \frac{1}{\rho_h} & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tilde{P} \qquad (5.18)$$

The intrinsic parameters are innate characteristics of the camera and sensor and comprise f, ρ_w, ρ_h, u_0 and v_0. The extrinsic parameters describe the camera's pose and comprise a minimum of six parameters to describe translation and orientation in

SE (3).

In our simulation environment:

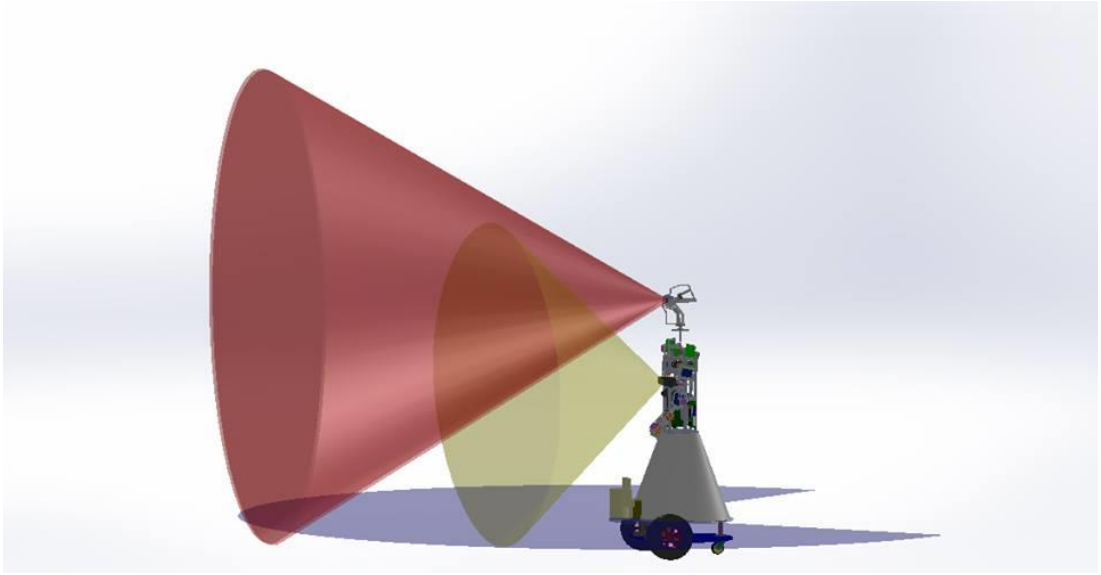


Figure 5.8 : UMAY's Field Of View(FOV).

f= 0.0080 m

$u_0 = 320$

$v_0 = 240$

$\rho_w = \ 1.0000\text{e} - 5$

$\rho_h = 1.0000\text{e-}5$

Perspective projection's derivative with respect to camera pose ξ_c is

$$p^{\cdot} = J_p(P, K, \xi_c)V \qquad (5.19)$$

Where V= $\left(v_x, v_y, v_z, \omega_x, \omega_y, \omega_z\right) \in IR^6$ is the velocity of the camera, the spatial velocity, which we introduced in advanced. $\boldsymbol{J_p}$ is a Jacobian-like object, but because we have taken the derivative with respect to a pose $\xi \in SE$ (3) rather than a vector it is technically called an *interaction matrix*. However in the visual servoing world it is more commonly called an *image Jacobian* or a *feature sensitivity matrix*.

Consider a camera moving with a body velocity $V = (v, \omega)$ in the world frame and observing a world point **P** with camera relative coordinates P = (*X*, *Y*, *Z*).

The velocity of the point relative to the camera frame is

$$\dot{p} = -\omega \times P - v \qquad (5.20)$$

as it was calculated in previous chapter we can write image jacobian in this form:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} -\frac{1}{z} & 0 & \frac{x}{z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{z} & \frac{y}{z} & 1+y^2 & -xy & -x \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \qquad (5.21)$$

Which relates camera velocity to feature velocity in normalized image coordinates. As it was explained, the normalized image-plane coordinates are related to the pixel coordinates by

$$u = \frac{f}{\rho_u} x + u_0 \qquad (5.22)$$

$$v = \frac{f}{\rho_v} y + v_0 \qquad (5.23)$$

Which we rearrange as

$$x = \frac{\rho_u}{f} \bar{u} \qquad (5.24)$$

$$y = \frac{\rho_v}{f} \bar{v} \qquad (2.25)$$

Where $\bar{u} = u - u_0$ and $\bar{v} = v - v_0$ are the pixel coordinates relative to the principal point. The temporal derivative is

$$\dot{x} = \frac{\rho_u}{f} \dot{\bar{u}} \qquad (5.26)$$

$$\dot{y} = \frac{\rho_v}{f} \dot{\bar{v}} \qquad (5.27)$$

And substituting Eq. (5.24-5.25) and Eq. (5.26-5.27) into Eq. (5.21) leads to

$$\begin{pmatrix} \dot{\bar{u}} \\ \dot{\bar{v}} \end{pmatrix} = \begin{pmatrix} -\frac{f}{\rho_u z} & 0 & \frac{\bar{u}}{z} & \frac{\rho_u \bar{u}\bar{v}}{f} & -\frac{f^2 + \rho_u^2 \bar{u}^2}{\rho_u f} & \bar{v} \\ 0 & -\frac{f}{\rho_v z} & \frac{\bar{v}}{z} & \frac{f^2 + \rho_v^2 \bar{v}^2}{\rho_v f} & -\frac{\rho_v \bar{u}\bar{v}}{f} & -\bar{u} \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \qquad (5.28)$$

In terms of pixel coordinates *with respect to the principal point*. We can write this in

concise matrix form as:

$$\dot{p} = J_p\, V \qquad (5.29)$$

Where J_p is the 2 × 6 *image Jacobian* matrix. Image Jacobians can also be derived for line and circle features, the matrix will be non-singular so long as the points are not coincident or collinear.in this work we use point feature method for 4 points so J_p is non-singular 4 × 6 image jacobian matrix.

$$\begin{pmatrix} \dot{\bar{u}} \\ \dot{\bar{v}} \end{pmatrix} = \begin{pmatrix} -\frac{f}{\rho_u z} & 0 & \frac{\bar{u}_1}{z} & \frac{\rho_u \bar{u}_1 \bar{v}_1}{f} & -\frac{f^2 + \rho_u^2 {\bar{u}_1}^2}{\rho_u f} & \bar{v}_1 \\ 0 & -\frac{f}{\rho_v z} & \frac{\bar{v}_1}{z} & \frac{f^2 + \rho_v^2 {\bar{v}_1}^2}{\rho_v f} & \frac{\rho_v \bar{u}_1 \bar{v}_1}{f} & -\bar{u}_1 \\ -\frac{f}{\rho_u z} & 0 & \frac{\bar{u}_2}{z} & \frac{\rho_u \bar{u}_2 \bar{v}_2}{f} & -\frac{f^2 + \rho_u^2 {\bar{u}_2}^2}{\rho_u f} & \bar{v}_2 \\ 0 & -\frac{f}{\rho_v z} & \frac{\bar{v}_2}{z} & \frac{f^2 + \rho_v^2 {\bar{v}_2}^2}{\rho_v f} & \frac{\rho_v \bar{u}_2 \bar{v}_2}{f} & -\bar{u}_2 \\ -\frac{f}{\rho_u z} & 0 & \frac{\bar{u}_3}{z} & \frac{\rho_u \bar{u}_3 \bar{v}_3}{f} & -\frac{f^2 + \rho_u^2 {\bar{u}_3}^2}{\rho_u f} & \bar{v}_3 \\ 0 & -\frac{f}{\rho_v z} & \frac{\bar{v}_3}{z} & \frac{f^2 + \rho_v^2 {\bar{v}_3}^2}{\rho_v f} & \frac{\rho_v \bar{u}_3 \bar{v}_3}{f} & -\bar{u}_3 \\ -\frac{f}{\rho_u z} & 0 & \frac{\bar{u}_4}{z} & \frac{\rho_u \bar{u}_4 \bar{v}_4}{f} & -\frac{f^2 + \rho_u^2 {\bar{u}_4}^2}{\rho_u f} & \bar{v}_4 \\ 0 & -\frac{f}{\rho_v z} & \frac{\bar{v}_4}{z} & \frac{f^2 + \rho_v^2 {\bar{v}_4}^2}{\rho_v f} & \frac{\rho_v \bar{u}_4 \bar{v}_4}{f} & -\bar{u}_4 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \qquad (5.30)$$

So far it was shown how points move in the image plane as a consequence of camera motion. To find *what camera motion is needed in order to move the image features at a desired velocity; we should find invers of jacobian matrix.*

$$V = \begin{pmatrix} J_{p1} \\ J_{p2} \\ J_{p3} \\ J_{p4} \end{pmatrix}^{-1} \begin{pmatrix} \dot{u}_1 \\ \dot{v}_1 \\ \dot{u}_2 \\ \dot{v}_2 \\ \dot{u}_3 \\ \dot{v}_3 \\ \dot{u}_4 \\ \dot{v}_4 \end{pmatrix} \qquad (5.31)$$

Given feature velocity we can compute the required camera motion, Considering **V** as the input to the robot controller, and if we would like for instance to try to ensure

an exponential decoupled decrease of the error

$\dot{e} = -\lambda e$

We obtain using:

$$\dot{P}^* = -\lambda(p^* - p) \qquad (5.32)$$

that *drives* the features toward their desired values p^* on the image plane. Combined with Eq. 5.31 and using the pseudo-inverse of jacobian matrix. We write

$$V = -\lambda \begin{pmatrix} J_{p1} \\ J_{p2} \\ J_{p3} \\ J_{p4} \end{pmatrix}^{-1} (p^* - p) \qquad (5.33)$$

In real visual servo systems, it is impossible to know perfectly in practice either J_p or J_p^{-1} . So an approximation or an estimation of one of these two matrices must be realized. In the sequel, we denote both the pseudo inverse of the approximation of the interaction matrix and the approximation of the pseudo inverse of the interaction matrix by the symbol $\widehat{J_p^{-1}}$. Using this notation, the control law is in fact:

$$V_e = -\lambda \widehat{J_p^{-1}} e \qquad (5.34)$$

Where $\widehat{J_p^{-1}}$ is the approximation of the pseudo-inverse of J_p .The output of the image-based controller is the reference velocity V_c of camera in eye in hand method. In this work, as we use eye to hand method we follow end effector and object in image plane of depth camera thus the output is Ve, velocity of end effector.

We have established the relationship between the velocity of individual joints and the translational and angular velocity of the robot's end-effector as follows**.** This equation describes the instantaneous forward kinematics where ν = (v_x, v_y, v_z, ω_x, ω_y, ω_z) ∈ R^6 is a spatial velocity and comprises translational and rotational velocity components. The matrix $J(q) \in R^{6\times N}$ is the manipulator Jacobian or the geometric Jacobian, as Humanoid robot UMAY is 6 DOF $J(q) \in R^{6\times 6}$.

$$V_e = J(q)\dot{q} \qquad (5.35)$$

After finding velocity of the end effector using suitable controller, we should find velocity of each joint using (5.35):

$$\dot{q} = J(q)^{-1} V_e \qquad (5.36)$$

In order to ensure better performances of a servoing system, different types of controller were used. Here we give a brief description of them.

5.6 Different Types Of Controller

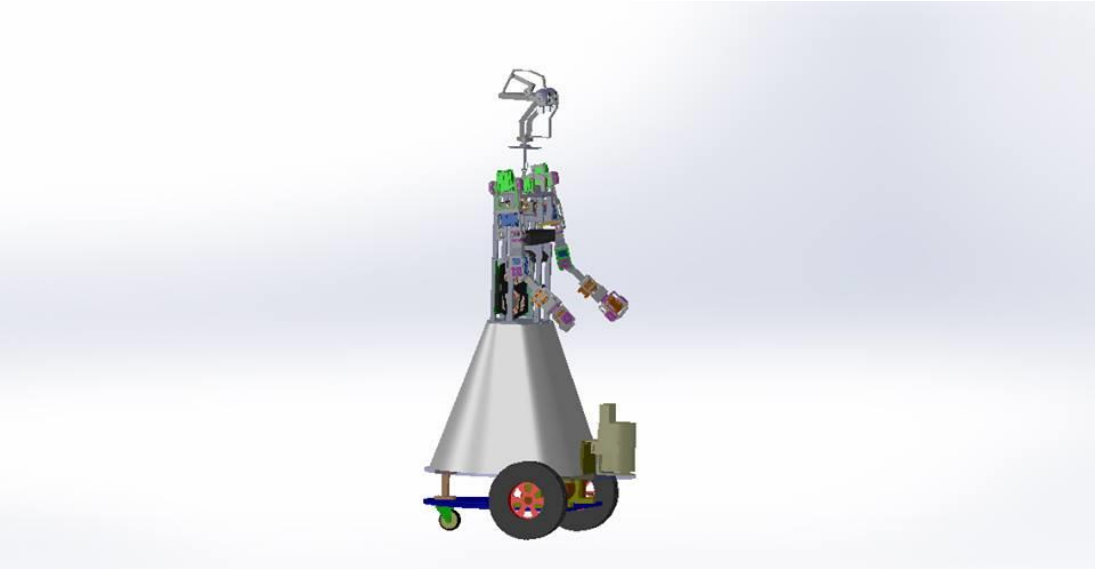


Figure 5.9 : UMAY's model.

5.6.1 Constant image jacobian:

Computing the image Jacobian requires knowledge of the camera intrinsic parameters, the principal point and focal length, but in practice it is quite tolerant to errors in these. It is a common alternative in using 2D cameras that is to define the true z in interaction matrix i.e. distance between initial position of end effector and desired object. This time the Jacobian is called the Constant Image Jacobian $J(p)^{\star}$. $J(p)^{\star}$ is constant over time and does not require image measurements for its adaptation to the current pose. In the simulations just discussed we have assumed that depth is known – this is easy in simulation but not so in reality. [peter corke]

5.6.2 Dynamical image jacobian:

The Jacobian also requires knowledge of Z_i, the distance to, or the depth of, each point. A number of approaches have been proposed to deal with the problem of

unknown depth. In case of using monocular pinhole camera the depth of the images point, distance between end effector and desired object are unknown, thus the real value of Jacobian is obscure.as we use kinect in this work the depth information of both end effector and desired object and therefore their distance from each other are known.so the value of interaction matrix $J(p)_n$ in each iteration can be calculated.

5.6.3 A mixed model

Malis proposes a way of constructing a mixed model, which consists of different linear approximations of the target function ϕ. Let x_n again be the current robot pose and x^* the teach pose. For a given robot command *u* we set again $\phi_n(u) := \phi(x_n, u)$ and now also $\phi^*(u) := \phi(x^*, u)$; In other words, both Image Jacobians, $Jn := J\Phi_n$ (0) and $J^\star := J\Phi\star$ (0) can be used as linear approximations of the behavior of the robot system. One of these models has its best validity at the current pose, the other at the teach pose. Since we are moving the robot from one towards the other it may be useful to consider both models. Malis proposes to use a mixture of these two models, i.e.[33]

$$y_{n+1} - y_n \approx \frac{1}{2}(J_n + J^*)u \qquad (5.37)$$

In his control law he calculates the pseudo inverse of the Jacobians, and therefore calls this approach "Pseudo-inverse of the Mean of the Jacobians", or short "PMJ". In a variation of this approach the computation of mean and pseudo-inverse is exchanged, which results in the "MPJ" method. The so-called "PMJ Controller" uses the pseudo-inverse of the mean of the two Jacobians Jn and $J^\star$. Using again a dampening factor $0 < \lambda \leq 1$ the controller output is given by:

$$u_n = \lambda . (\frac{1}{2}(J_n + J^*))^{-1}\Delta y_n \qquad (5.38)$$

Analogously, the "MPJ Controller" works with the mean of the pseudo-inverse of the Jacobians:

$$u_n = \lambda . (\frac{1}{2}({J_n}^{-1} + {J^*}^{-1}))\Delta y_n \quad (5.39)$$

This controller will drive the end effector so that Umay's arm moves toward the desired position. It is important to note that nowhere have we required the pose of the

camera or of the object, everything has been computed in terms of what can be measured on the image plane.

5.7 Simulation Results

In this section, the simulation results on the proposed image-based stereo visual servoing system and the performances of the system for both eye in hand and eye to hands systems are presented, two different tasks are tested with a kinect as a sample of depth camera visual servoing system. Using one of the models defined above we wish to design a controller which steers the robot arm towards an object of unknown pose.

One of our challenges in this project was finding a program that can simulate kinect, programs can simulate depth camera are rare. In this work a software platform ROS for analysis and design of robotic systems to simulate kinematics of robotic structures was presented .It was developed within the gazebo environment to simulate Kinect devices.it allows implementing control strategies in order to follow trajectories, perform tasks, etc. Thus it is very suitable to implement robotic experiments before dealing with the real system.

5.7.1 Simulation result for controller using constant image jacobian:

In Simulation the complete camera-robot system is modeled .The control problem can be expressed in terms of image coordinates.

The task is to move end effector indicated by green square toward desired point indicated by red square. The points may, but do not have to, follow the straight-line paths indicated by dot line.

Figure 5.10 shows the Views of end effector trajectory using controller with Constant Image Jacobian toward desired point.as you can see the end effector indicated by green square and desired point indicated by red square.As the controller is constant image jacobian so we use true z in interaction matrix then velocity of end effector toward desired point is constant .You can see the trajectory of end point in Figure 5.10 , 5.12, 5.14, 5.16 trajectory indicated by white dots toward red desired point.Fugure 5.11 ,5.13, 5.15, 5.17 show the velocity of end effector toward desired point .

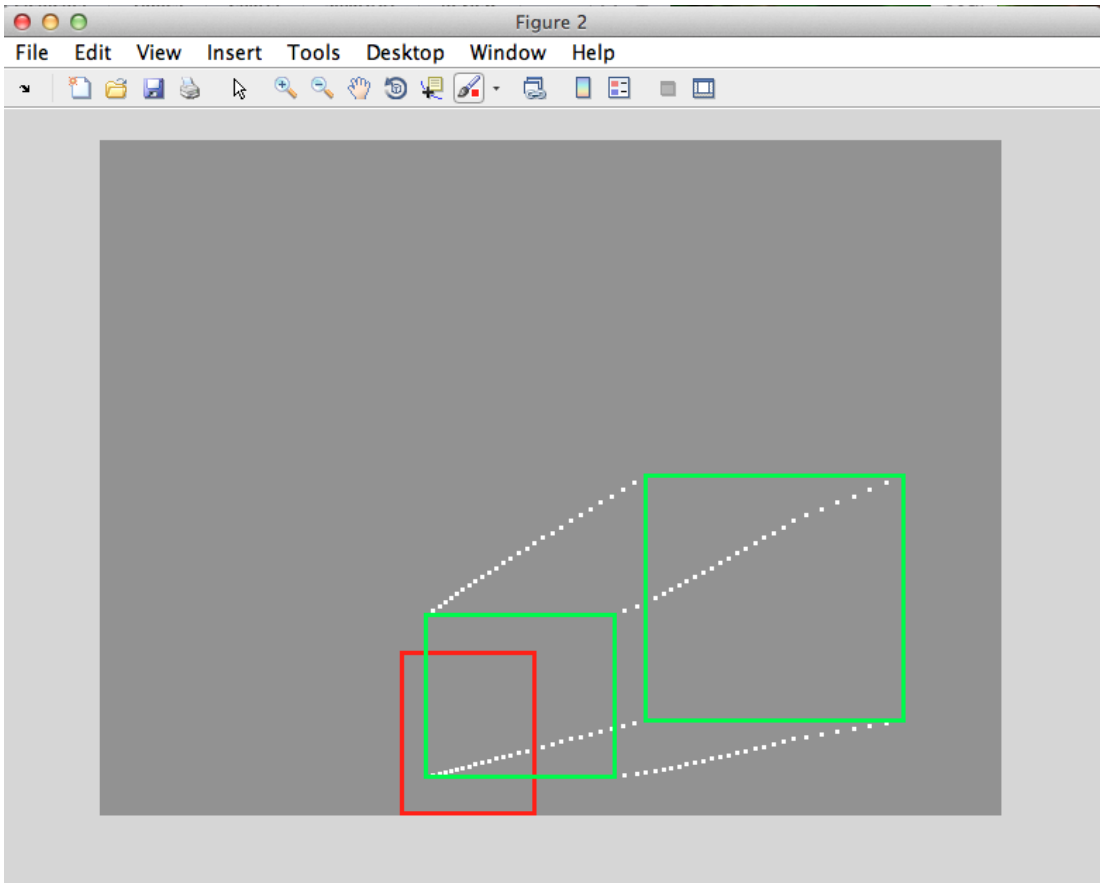


Figure 5.10 : Views of end effector trajectory using controller with Constant Image Jacobian.

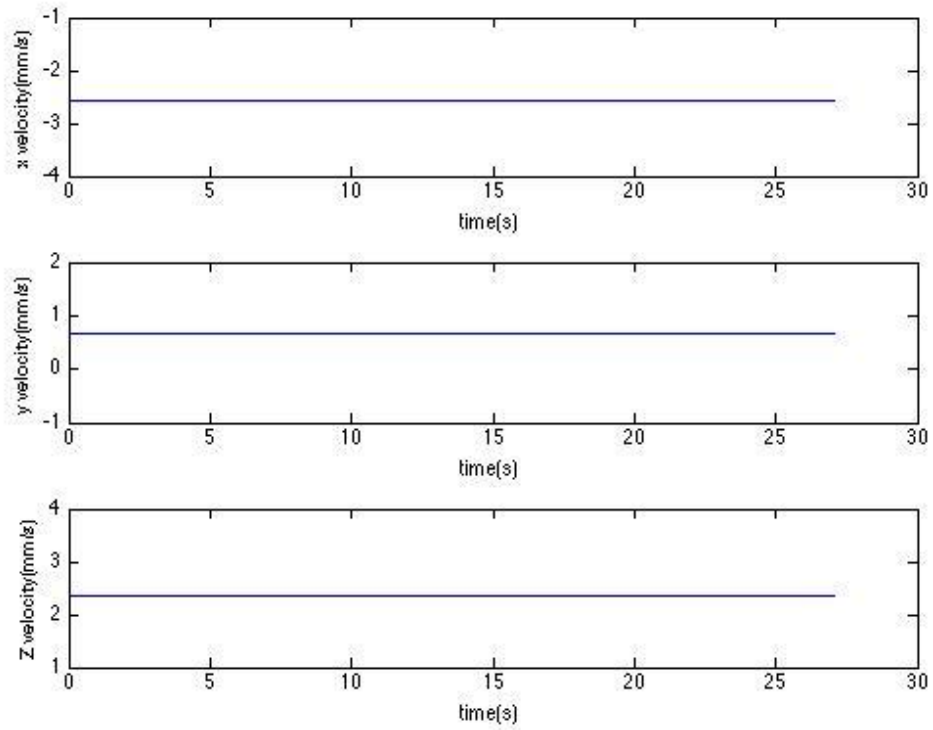


Figure5.11 : velocity of end effector with controller using constant Image Jacobian.

5.7.2 Simulation result for controller using dynamical image jacobian:

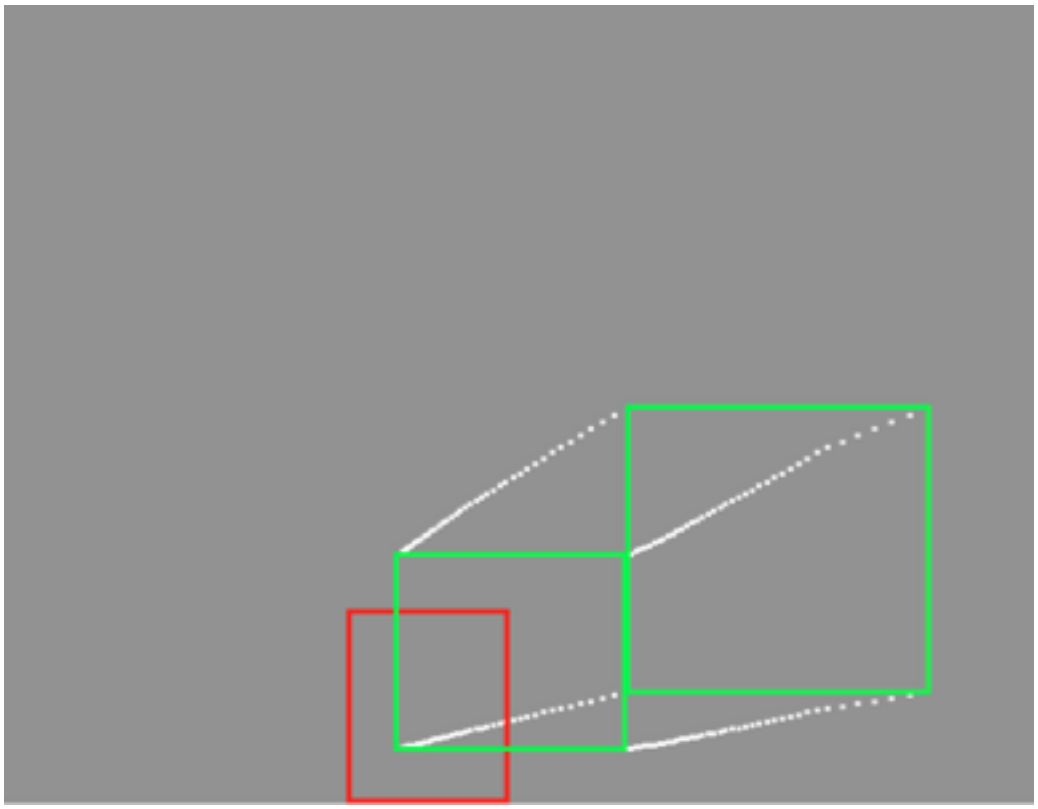


Figure5.12 : Views of end effector trajectory toward desired point with Dynamical Z.

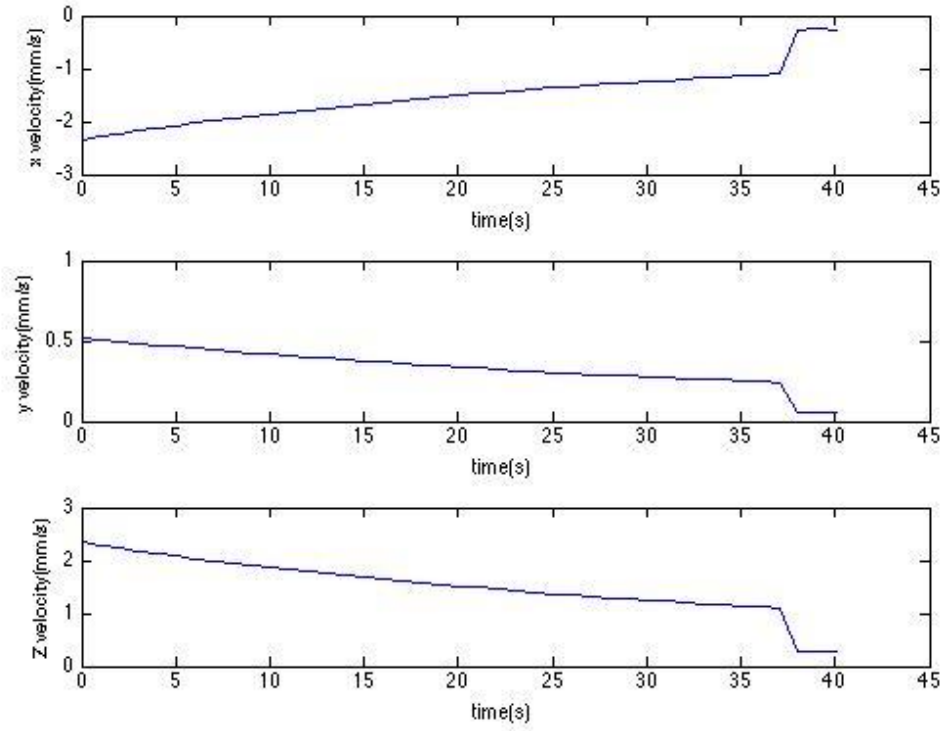


Figure5.13 : Velocity of end effector with controller using Dynamical Image Jacobian.

5.7.3 Simulation result for controller using PMJ image jacobian:

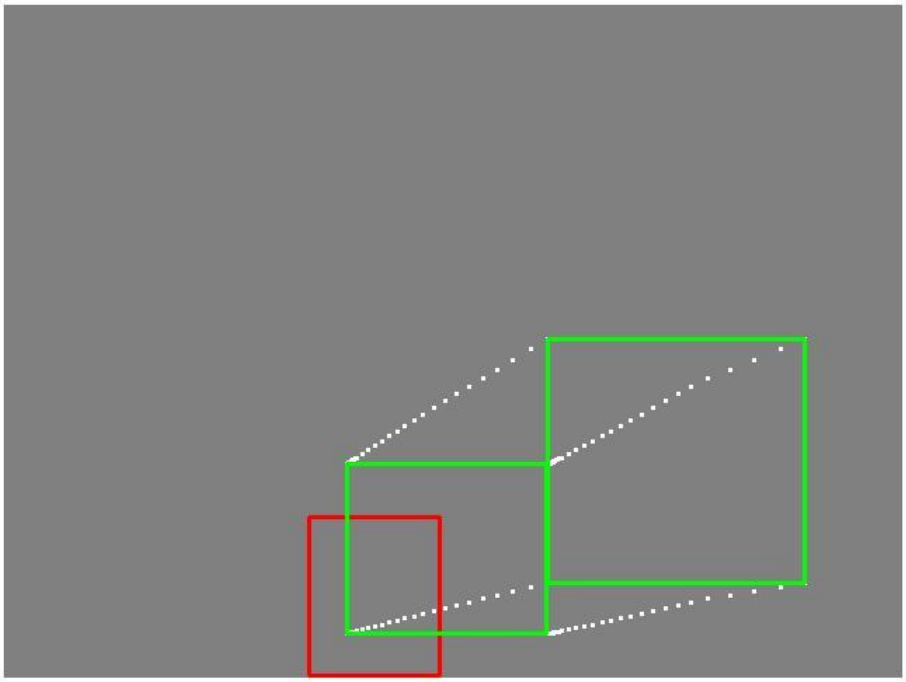


Figure5.14 : Views of end effector trajectory toward desired point with controller using PMJ.

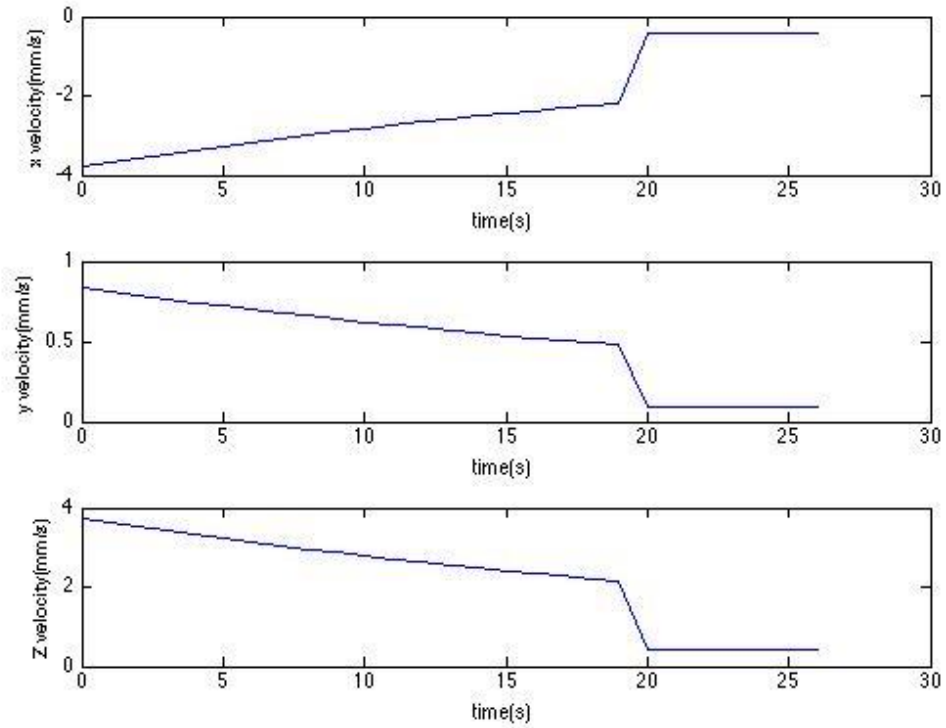


Figure5.15 : Velocity of end effector with controller using Mixed model(PMJ) Image Jacobian.

5.7.4 Simulation result for controller using MPJ image jacobian:

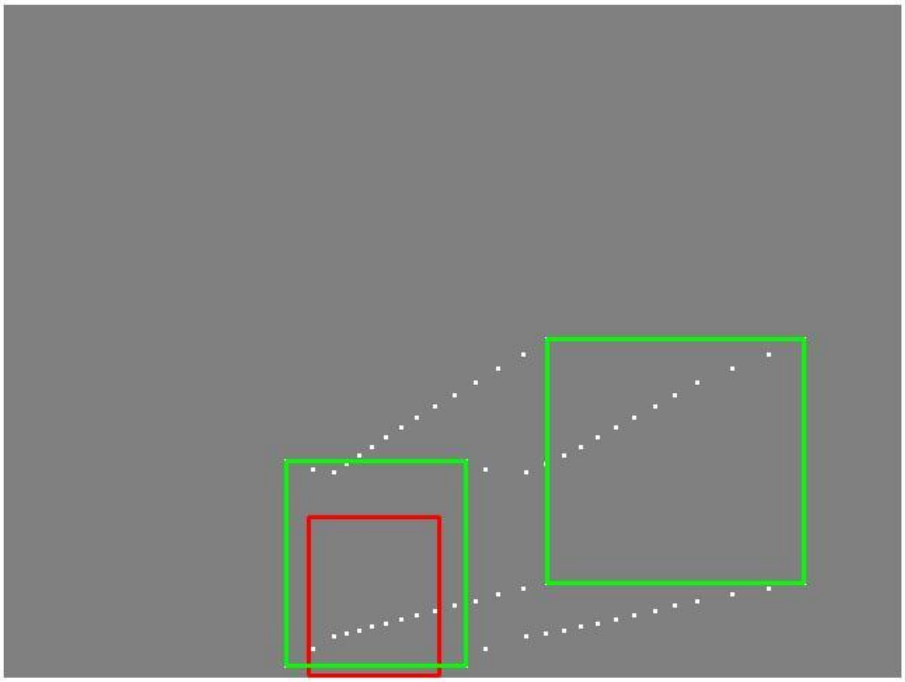


Figure5.16 : Views of end effector trajectory toward desired point with MPJ controller.

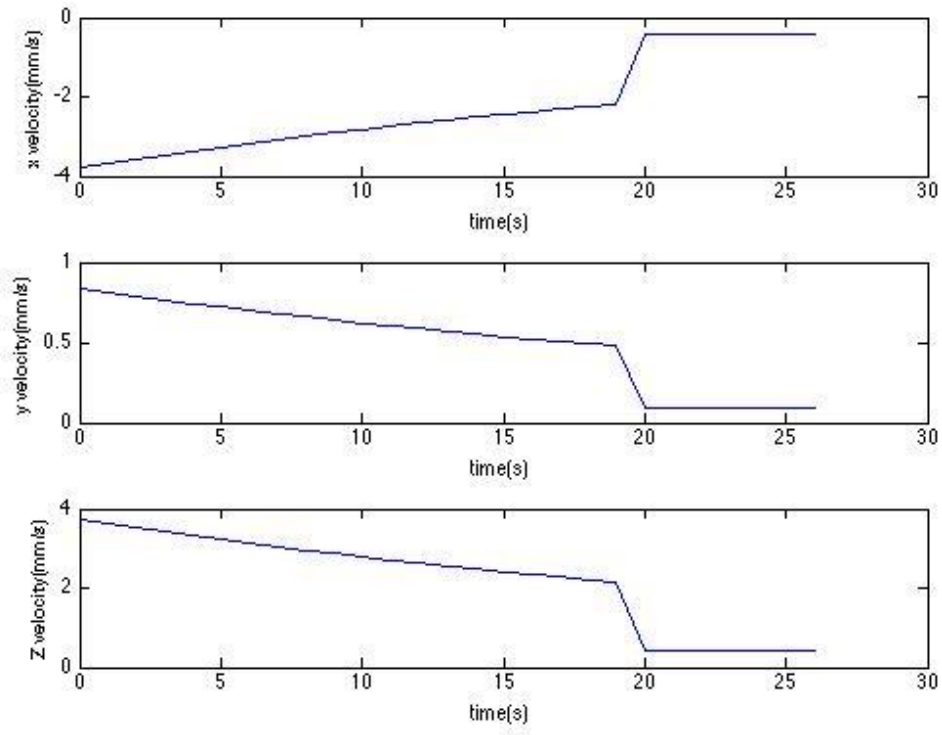


Figure 5.17 : Velocity of end effector with controller using Mixed model(MPJ) Image Jacobian.

As it was saied before Desired point is indicated by red square and end effector indicated by green square, the trajectory of end effector toward desired point shown by white dots.

In case of Dynamical Image jacobian controller and MPJ and PMJ controller we get depth information of both end effector and desired point in each iteration.so the distance between end effector and desired point can be calculated and substutude in interaction matris .

In contrast to constant image jacobian which has constant velocity; dynamical Image jacobian and MPJ and PMJ controllers give us variable velocity. You can see them on results parts.

As you see above end effector reaches the desire point, According to the given results, it can be said that the controller with Dynamical Image Jacobian ,gives better result in compare to common constant z controllers.

Even though end effector come close and even reach to the desired object,in all methods but it can be seen in figure that shows the trajectory of the MPJ controller, The MPJ method is the clearly the winner in this comparison.

5.7.5 Screenshot of the simulation environment

In Simulation the complete camera-robot system is modeled. This includes the complete UMAY robot arm with inverse kinematics, rendering of the camera image in a realistic resolution and application of the image processing algorithms to obtain the image features.

By using image processing algorithm camera can recognized end effector (green square) and desired point (red square).

As you can see we have two views one of them is view of depth camera (kinect)

And the other is view of whole system.

view of kinect shows us both end effector and desired point from near distance in more details.

In view of whole system we can see motion of robot arm toward desired object

See Figure 5.18 , 5.19.

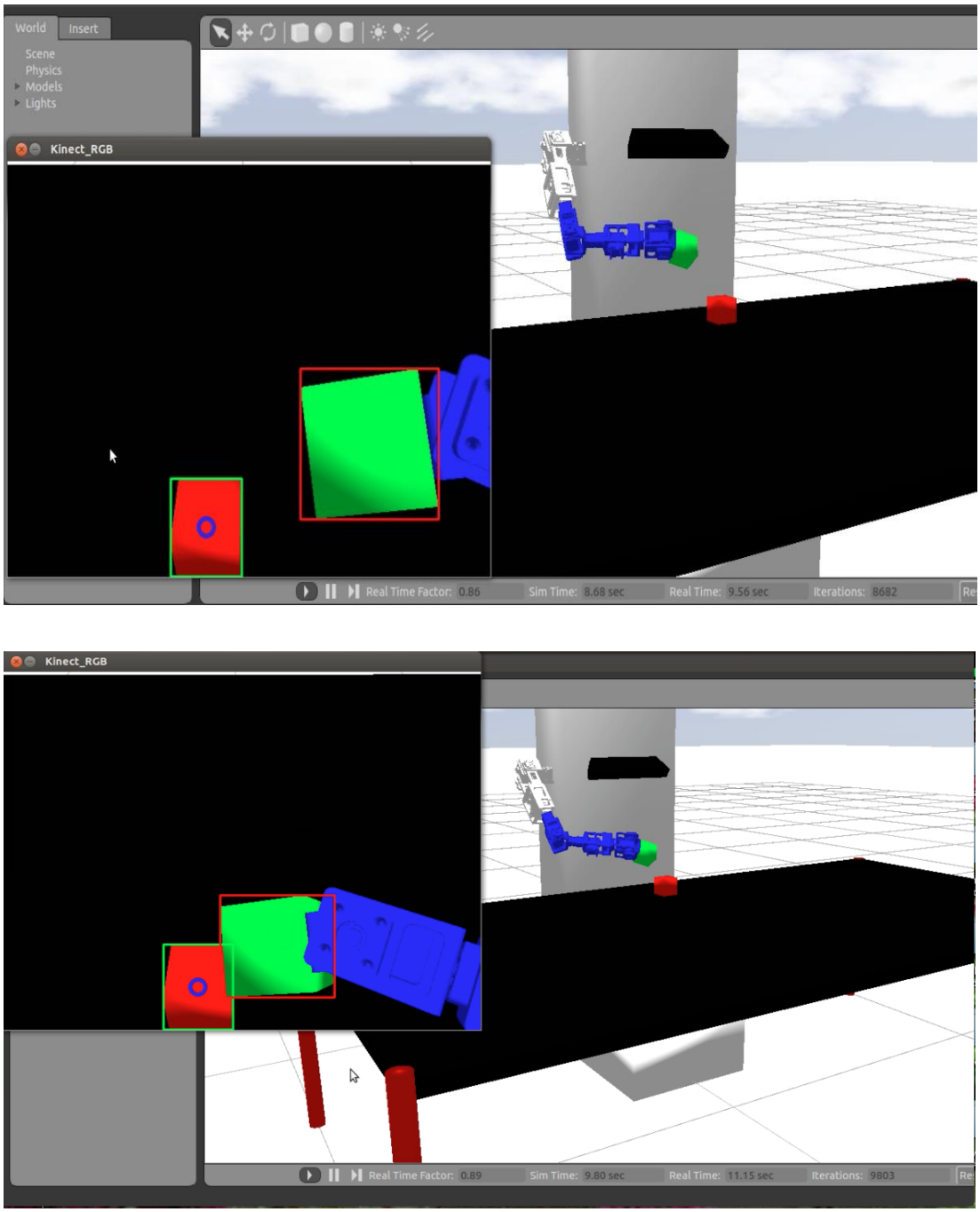


Figure 5.18 : Screenshot of the simulation environment.

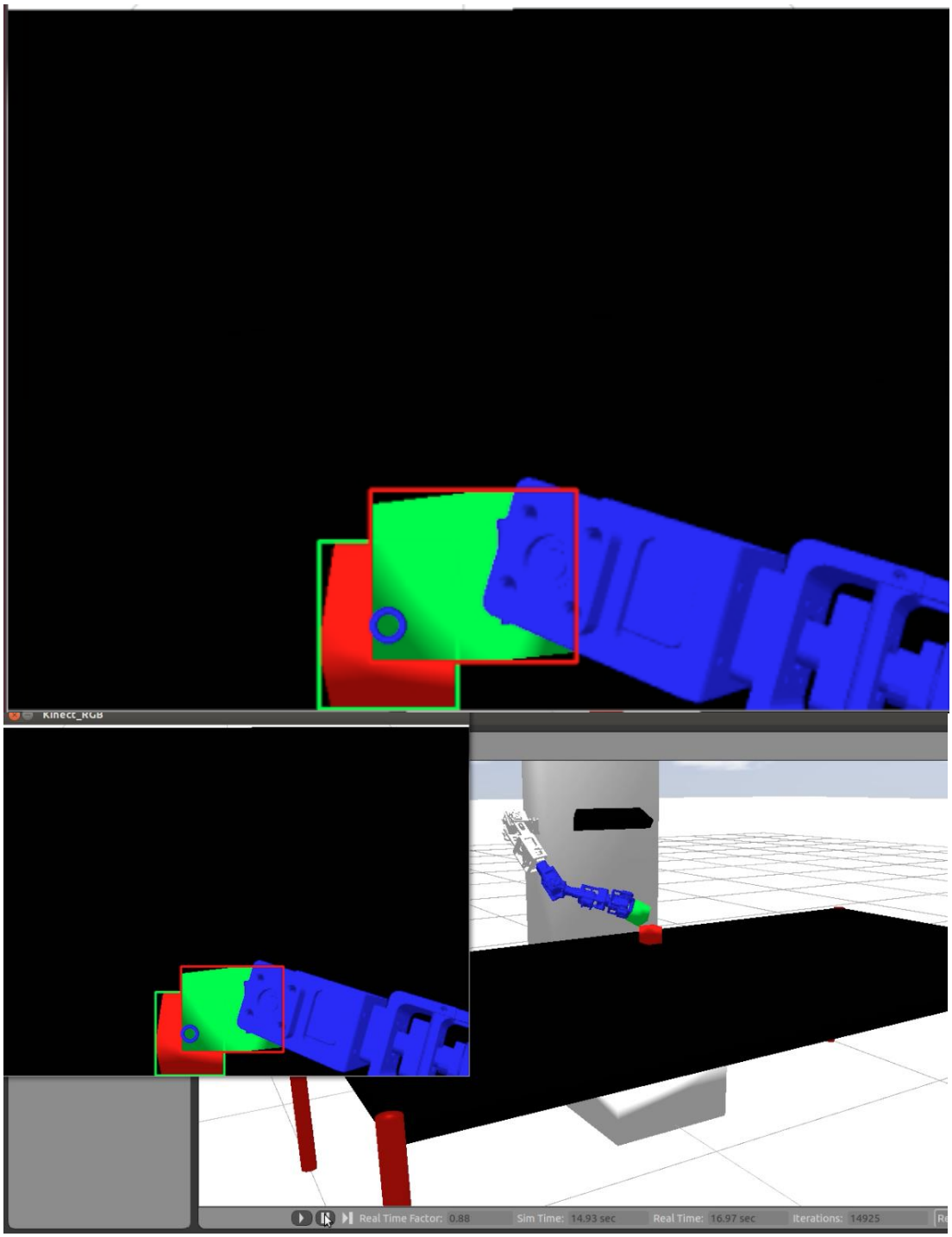


Figure 5.19 : motion of robot arm toward desired object.

REFERENCES

[1] **P. Corke** (1996). *Visual control of robots : high-performance* visual servoing vol. 2. Taunton England ; New York, NY: Research Studies Press ; Wiley

[2] **J. T. Feddema and O. R. Mitchell**. (1986)."Vision-guided servoing with feature-based trajectory generation [for robots]," *Robotics and Automation, IEEE Transactions on,* vol. 5, pp. 691-700

[3] **B. Espiau, *et al*** . . (1992)."A new approach to visual servoing in robotics," *Robotics and Automation, IEEE Transactions on,* vol. 8, pp. 313-326

[4] **Faunc-Robotics..** (2009). *FANUC Robotics Industrial Robots and Robotic Automated Systems*.

[5] **Robot-Worx. .** (2008). *Robot Use in the Present and Future*.

[6] **W. Guo-Qing, *et al*** (1997). "Real-time visual servoing for laparoscopic surgery. Controlling robot motion with color image segmentation," *Engineering in Medicine and Biology Magazine, IEEE,* vol. 16, pp. 40-45.

[7] **D. B. Samadi.** (2012). *da Vinci Si robotic system wants to be your surgeon*.

[8] **S. S. Mehtatt, *et al.*** (2006). "Visual servo control of an unmanned ground vehicle via a moving airborne monocular camera," in *American Control Conference,* p. 6 pp.

[9] **Macroswiss.** (2012). *Macroswiss Unmanned Ground Vehicles*.

[10] **S. Ye, *et al***. (2012). "Study on intelligent visual servoing of space robot for cooperative target capturing," in *Information and Automation (ICIA), International Conference on*, 2012, pp. 733-738.

[11] **Kozima, Hideki, Nakagawa, Cocoro and Yasuda, Yuriko.** (2005). Interactive Robots for Communication-Care: A Case-Study in Autism Therapy. *IEEE Internations Workshop on Robots and Human Interactive Communication.*

[12] . **Colton, M., et al.** (2009). *Toward therapist-in-the-loop assistive robotics for children with autism and specific language impairment***et** al. s.l. : AISB Symposium: New Frontiers in Human-Robot Interaction.

[13] **Don Joven Agravante , François Chaumette**. (2013).Visual Servoing for the REEM Humanoid Robot's Upper Body. Pages 2, ,IEEE Int. Conf. on Robotics and Automation, ICRA'13 5233-5238

[14] **R. Horaud, F. Dornaika, C. Bard, and B. Espiau.** (1995). "Visually guided object grasping," IEEE Transactions on Robotics and Automation, vol. 14, pp. 525–532.

[15] **D. Kragic and H. I. Christensen. (**2002)"Model based techniques for robotic servoing and grasping," in IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'02, (Lausanne, Switzerland), pp. 299 – 304.

[16] **P. Boyraz, C.Bora Yigit, B. H.Okan**. . (2013) "UMAY: A Modular Humanoid Platform For Education and Rehabilitation of Childran with Autism Spectrum Disorder", IEEE International Conference on Control (ASCC),2013 9th Asian, Istanul, Turkey, 23-26 June

[17] **S. Hutchinson, G. D. Hager, and P. I. Corke.** (1996)"A tutorial on visual servo control," IEEE Trans. Robot. Automat., vol. 12, no. 5, pp. 651–670, Oct. 1996.

[18] **F. Chaumette and S. Hutchinson.** (2006) "Visual servo control. part I: Basic ap- proaches," IEEE Robot. Automat. Mag., vol. 13, no. 4, pp. 82–90, Dec.

[19] **F. Chaumette and S. Hutchinson.** (2007) "Visual servo control. part II: Advanced approaches [tutorial]," IEEE Robot. Automat. Mag., vol. 14, no. 1, pp. 109–118, Mar.

[20] **D. Kragic and H. I. Christensen.** (2002) "Survey on visual servoing for manipulation," Royal Institute of Technology, ISRN KTH/NA/P–02/01–SE, Tech. Rep. CVAP 259, January.

[21] **Y. Shirai and H. Inoue.** (1973) "Guiding a robot by visual feedback in assembling tasks," *Pattern Recognition,* vol. 5, pp. 99-108.

[22] **S. I. A. I. Center and G. J. Agin**. (1979). *Real Time Control of a Robot with a Mobile Camera*: SRI International.

[23] **P. Corke.** (1993) "Visual Control of Robot Manipulators - A Review," in *Visual Servoing*, pp. 1-31.

[24] **A. C. Sanderson and L. E. Wciss.** (1980) "Image-based visual servo control using relational graph error signals," *Proc. IEEE,* pp. 1074-1077.

[25] **Azad Shademan.** (2013) International Journal of Computer Vision (IJCV) Special Issue: Vision and Robotics". - Computing Science, University of Alberta

[26] **L. E. Weiss, A. C. Sanderson, and C. P. Neuman.** (1987) "Dynamic sensor based control of robots with visual feedback," IEEE Trans. Robot. Automat., vol. 3, no. 5, pp. 404–417, October 1987.

[27] **F. Chaumette.** (2004). "Image moments: a general and useful set of features for visual servoing," IEEE Trans. Robot., vol. 20, no. 4, pp. 713–723, Aug. 13, 16, 17, 25, 45

[28] **G. Hager and K. Toyama.** (1998) "XVision: A portable substrate for real-time vision applications," Computer Vision and Image Understanding, vol. 69, no. 1, pp. 23–37, January.

[29] **W. J. Wilson, C. C. W. Hulls, and G. S. Bell**. (1996).“Relative end-effector control using cartesian position based visual servoing,” IEEE Trans. Robot. Automat., vol. 12, no. 5, pp. 684–696, October 13, 14

[30] **B. Espiau, F. Chaumette, and P. Rives**. (1992) “A new approach to visual servoing in robotics,” IEEE Trans. Robot. Automat., vol. 8, no. 3, pp. 313–326, June 1992. 13, 14, 16, 25

[31] **L. Deng, F. Janabi-Sharifi, and W. J. Wilson**. (2005) “Hybrid motion control and planning strategies for visual servoing,” IEEE Trans. Ind. Electron., vol. 52, no. 4, pp. 1024–1040.

[32] **N. R. Gans and S. A. Hutchinson** (2007) “Stable visual servoing through hybrid switched-system control,” IEEE Trans. Robot., vol. 23, no. 3, pp. 530–540, 13

[33] **E. Malis and P. Rives**. (2003).“Robustness of image-based visual servoing with respect to depth distribution errors,” in Proc. IEEE Int. Conf. Robot. Automat., vol. 1, Sept. 14–19, pp. 1056–1061. 13

[34] **V. Kyrki.** (2009) “Control uncertainty in image-based visual servoing,” in Proc. IEEE Int. Conf. Robot. Automat., May 12–17, 2009, pp. 1516–1521. 13

[35] **N. Gans, S. A. Hutchinson, and P. I. Corke**. (2003)“Performance tests for visual servo control systems, with application to partitioned approaches to visual servo control,” Int. J. Robot. Res., vol. 22, no. 10–11, pp. 955–981, 13

[36] **Marco Antonio P´erez Cisneros**. (2004).INTELLIGENT MODEL STRUCTURES IN VISUAL SERVOING ,by, September.

[37] **L.E. Weiss, A.C. Sanderson, and C.P. Newman.** (1987) Dynamic sensor-based control of robots with visual feedback. IEEE Transactions on Robotics and Automa- tion, 3(5):404–417, October.

[38] **G.D. Hager, S. Hutchinson, and P. Corke.** (1996) Tt3: Tutorial on visual servo control, workshop notes. IEEE International Conference on Robotics and Automation.

[39] **P.I. Corke and M.C. Good.** (1996).Dynamic effects in visual closed-loop systems. IEEE Transactions on Robotics and Automation, 12(5):671–683, October

[40] **G. Flandin, *et al*.** (2000)**.** "Eye-in-hand/eye-to-hand cooperation for visual servoing," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*. pp. 2741-2746 vol.3.

[41] **G. Chesi and K. Hashimoto.** (2002) "Static-eye against hand-eye visual servoing," in *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*,pp. 2854-2859 vol.3. [1]

[42] **F. Chaumette.** (1998)."Potential problems of stability and convergence in image-based and position-based visual servoing The confluence of vision and control." vol. 237, D. Kriegman, *et al.*, Eds., ed: Springer Berlin / Heidelberg, pp. 66-78.

[43] **J. Armstrong Piepmeier.** (2002)., "Uncalibrated eye-in-hand visual servoing," in *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, pp. 568-573 vol.1.

[1] **Abrahart, R. J., and See, L.** (1998). Neural Network vs. ARMA Modelling: Constructing Benchmark Case Studies of River Flow Prediction. In GeoComputation ’98. Proceedings of the Third International Conference on GeoComputation, University of Bristol, United Kingdom, 17–19 September (CD-ROM).

[44] **E. Cervera.** (2002). "Is 3D useful in stereo visual contro*l*?," in Robotics and Automation, Proceedings. ICRA '02. IEEE International Conference on, 2002, pp. 1630-1635 vol.2.

[45] **D.-J. Kim*, et* al.** (2009). "Eye-in-hand stereo visual servoing of an assistive robot arm in unstructured environments," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pp. 2326-2331.

[46] **R. Mahony, et al.** (2002). "Choice of image features for depth-axis control in image based visual servo control," presented at the IEEE International Conference on Intelligent Robots and Systems, 2002.

[47] **M. A. Arlotti and M. N. Granieri.** (1991). "A perception technique for a 3-D robotic stereo eye-in-hand vision system," in *Advanced Robotics, 'Robots in Unstructured Environments', 91 ICAR., Fifth International Conference on*, pp. 1626-1629 vol.2.

[48] **A. C. Sanderson and L. E. Weiss**. (1980) "Image Based Visual Servo Control Using Relational Graph Error Signal," presented at the IEEE International Conference of Cybernetics and Society.

[49] **Mohammad Keshmiri, Mehdi Keshmiri, Abolfazl Mohebbi** (2010) ,Augmented online point to point trajectory planning, a new approach in catching a moving object by a manipulator , Control and Automation (ICCA), 8th IEEE International Conference.

[50] **Abolfazl Mohebbi.** (2012).Augmented Image Based Visual Servoing of a Manipulator using Acceleration Command , IEEE Transactions on Industrial Electronics 10

[51] **Mohammad Keshmiri, Wen-Fang Xie, Abolfazl Mohebbi.** (2013). Augmented image based visual servoing of a Manipulator using acceleration command , IEEE Transactions on Industrial Electronics 11

[52] **K. Hashimoto and T. Noritsugu.** (1998)."Performance and sensitivity in visual servoing," in *Robotics and Automation, Proceedings. 1998 IEEE International Conference on*, 1998, pp. 2321-2326 vol.3.

[53] **S. Hutchinson.** (1996) *et al.*, "A tutorial on visual servo control," *Robotics and Automation, IEEE Transactions on*, vol. 12, pp. 651-670.

[54] **W. J. Wilson.** (1996) "Relative end-effector control using Cartesian position based visual servoing," *Robotics and Automation, IEEE Transactions on*, vol. 12, pp. 684-696.

[55] **D. G. Lowe.**. (1987) "Three-dimensional object recognition from single two-dimensional images," *Artif. Intell.*, vol. 31, pp. 355-395.

[56] **G. Chesi and K. Hashimoto**. (2004)."A simple technique for improving camera displacement estimation in eye-in-hand visual servoing," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 26, pp. 1239-1242.

[57] **F. Chaumette and S. Hutchinson.** (2006) "Visual servo control. I. Basic approaches," *Robotics & Automation Magazine, IEEE,* vol. 13, pp. 82-90.

[58] **L. Weiss and A. Sanderson.** (1987) "Dynamic sensor-based control of robots with visual feedback," *IEEE Journal of Robotics and Automation,* vol. 3, pp. 404-417.

[59] **P. I. Corke and SpringerLink.** (2011). *Robotics, vision and control fundamental algorithms in MATLAB®.*

[60] **F. Chaumette and S. Hutchinson**. (2007)."Visual Servo Control, Part II: Advanced Approaches," *IEEE Robotics and Automation Magazine,* vol. 14, pp. 109-118.

[61] **F. Chaumette and E. Malis.** (2000). "2 1/2 D visual servoing: a possible solution to improve image-based and position-based visual servoings," in Robotics and Automation,Proceedings. *ICRA '00. IEEE International Conference.*

[62] **R. Horaud, F. Dornaika, and B. Espiau.** (1998).Visually guided object grasping. Robotics and Automation, IEEE Transactions on, 14(4):525–532.

[63] **P. Martinet and E. Cervera.** (2001)."Stacking Jacobians properly in stereo visual servoing," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on,*pp. 717-722 vol.1.

[64] **Hartley, R.~I. and Zisserman, A.** (2000). "Multiple View Geometry in Computer Vision", publisher: "Cambridge University Press, ISBN: 0521623049"

[65] **O.Arif,W.Daley,P.A.Vela,J.Teizer,andJ.Stewart.** (2010).Visual tracking and segmentation using time-of-flight sensor. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 2241 –2244, sept.

[66] **A.BleiweissandM.Werman.** (2009).Fusing time-of-flight depth and color forreal-time segmentation and tracking. In *Proceedings of DAGM 2009 Workshop on Dynamic 3D Imaging*, pages 58– 69.

[67] **S. May, B. Werner, H. Surmann, and K. Pervolz.** (2006).3d time-of-flight cameras for mobile robotics. In *Proc. IEEE/RSJ Conf. on Intelligent Robots and Systems*, pages 790–795.

[68] **P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. Rgbd.** (2010) mapping: Using depth cameras for dense 3d modeling of indoor environments. *In RGB-D: Advanced Reasoning with Depth Cameras Workshop in conjunction with RSS*.

[69] **J. Smisek, M. Jancosek, and T. Pajdla.** (2011)3d with kinect. In *Proc. ICCV Workshops*, pages 1154–1160.

[70] **B. Freedman, A. Shpunt, M. Machline, and Y. Arieli.** (2012).Depth mapping using projected pat- terns.

[71] **S.Soutschek,J.Penne,J.Hornegger,andJ.Kornhuber.** (2008).3-dgesture-basedscenenavigationin medical imaging applications using time-of-flight cameras. In *Proc. CVPR 2008 Workshops*, pages 1–6.

[72] **R. Szeliski.** (2010).*Computer Vision: Algorithms and Applications*. New York: Springer.

[73] **R. Hartley and A. Zisserman.** (2004).*Multiple View Geometry in Computer Vision*. Cambridge Uni- versity Press.

[74] **J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake.** (2011).“Real-Time Human Pose Recognition in Parts from Single Depth Images,” in *CVPR*, 2011.

[75] **Willow garage turtle bot.** http://www.willowgarage.com/turtlebot.

[76] **S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, “KinectFusion**(2011). real-time 3D reconstruction and interaction using a moving depth camera,” in *Proceedings of the 24th annual ACM Symposium on User Interface Software and Technology*.

[77] **J. Smisek, M. Jancosek, and T. Pajdla** .(2011)“3d with kinect,” in *1st IEEE Workshop on Consumer Depth Cameras for Computer Vision*.

[78] **K.Konoldige and P.Mihelich**. (2011).“Technical description of kinect calibration,”tech. rep.,Willow Garage, http://www.ros.org/wiki/kinect calibration/technical.

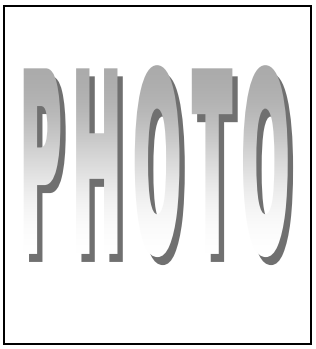
[79] **J. Salvi, J. Pags, and J. Batlle.** (2004). “Pattern codification strategies in structured light systems,” *Pattern Recognition*, vol. 37, pp. 827–849

[80] **J. Park** (2007)."Trajectory estimation of a moving object using Kalman filter and Kohonen networks," *Robotica*, vol. 25, pp. 567-574.

[81] **Elizabeth Alexander.** (2011).Affordable compact humanoid robot for autism spectrum disorder in children.

[82] **Mark W.Spong (Author), Seth Hutchinson.** (2005) (Author) book .Robot Modeling and Control

CURRICULUM VITAE



Name Surname: Arezou Rahimi

Place and Date of Birth: Tabriz/Iran, 21.09.1986

Address: Istanbul Technical University, Prof. Dr. Ali Ihsan Aldogan dormitory

E-Mail: arahimi@itu.edu.tr

B.Sc.: **Electronic Engineering**

List of Publications and Patents:

- Hosseini B, Yigit C. B, Rahimi A , Boyraz P, (2013) Indirect Force Control in 6 DOF Humanoid Robot arm using impedance controller CONTECH 2013, Aralik 26-28, 2013, Istanbul, Turkey.

PUBLICATIONS/PRESENTATIONS ON THE THESIS

- Rahimi A , Yigit C. B, Hosseini B , Boyraz P, 2013: Visual Servo-Control Application in Humanoid Robot using Depth Camera İnformation, CONTECH 2013, Aralik 26-28, 2013,Istanbul, Turkey.