TECHNISCHE UNIVERSITÄT CHEMNITZ

Development of Integration Algorithms for Vision/Force Robot Control with Automatic Decision System

Von der Fakultät für Elektrotechnik und Informationstechnik
der Technischen Universität Chemnitz

genehmigte

Dissertation

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

vorgelegt von

Dipl.-Ing. Mohamad Bdiwi
geboren am 25. August 1985 in Aleppo, Syrien

eingereicht am

20.01.2014

Gutachter

Prof. Dr.-Ing. Jozef Suchý
Prof. Dr.-Ing. Michael Gerke

Tag der Verleihung

Contents

List of Symbols and Notation

x	A scalar value
$\hat{x}$	An estimated value
$x_i x_j$	A segment of a line
$X, \vec{X}$	A vector
X^T	The transpose of X
$\dot{X}$	A velocity vector
${}^{C}X$	A pose relative to the camera coordinate system
${}^{W}X$	A pose relative to the word coordinate system
${}^{T}X$	A pose relative to the tool coordinate system
${}^{S}X$	A pose relative to the visible side coordinate system
${}^{W}T_C$	A transformation matrix from camera to world coordinate system
X_m	A measured pose
X_d	A desired pose (commanded by the user)
X_v	A desired pose calculated from vision system
X_f	A desired pose calculated from force error
$\Delta\vec{X}$	A pose error
$\Delta\vec{X}_S$	A pose error after multiplication with the selection matrix
$\vec{q}$	The pose of robot joints
$\dot{\vec{q}}$	The velocity of robot joints
J	A Jacobian matrix
J^{-1}	The inverse of Jacobian matrix
$\vec{F}_d$	A desired force vector

$\vec{F}_m$	A measured force vector
$\vec{S}_d$	A desired vector of visual features
$\vec{S}_m$	A measured vector of visual features
(u, v, w)	The axes of image coordinate system
(x, y, z)	The axes of Cartesian coordinate system
$M[x, y]$	A 2D matrix, where x number of rows and y number of columns
$M(i, j)$	A scalar value of the element (i, j) in the matrix M
$\{a, b, c\}$	A set of individual elements

List of Acronyms

SIFT	Scale Invariant Feature Transform
SURF	Speed-Up Robust Features
IBVS	Image Based Visual Servoing
PBVS	Position Based Visual Servoing
ECL	End-effector Close Loop
EOL	End-effector Open Loop
RGB	Red Green Blue
RGBD	Red Green Blue Depth
HSV	Hue Saturation Value
S matrix	Selection Matrix
RFID	Radio Frequency Identification
HRI	Human Robot Interaction
PHRI	Physical Human Robot Interaction
CCD	Charge Coupled Device
CMOS	Complementary Metal Oxide Semiconductor
IR	Infra-Red
FTS	Force Torque Sensor
OCR	Optical Character Recognition

HT	Hough Transform
PPHT	Progressive Probabilistic Hough Transform
AOI	Area Of Interest

Chapter 1

Introduction

1.1 Motivation

Advanced robot researches are growing worldwide rapidly and they are providing the humanity with sophisticated systems. In accordance to that, people expect more and more from robots to perform different applications and tasks which need physical interaction between the robot, the environment and the human, such as taking care of elderly and disabled people or even as assistance robots helping the workers in the factories or in the industrial applications. Based on the results of the International Federation of Robotics statistics (IFR, 2012), IFR has estimated the following:

- More than 160,000 industrial robots were sold in 2012.
- About 2.5 million service robots for personal and domestic were sold in 2011. So far, service robots for personal and domestic use are mainly in the areas of domestic household robots (e.g. vacuum and floor cleaning), lawn-mowing robots, humanoid robot, entertainment robots, education robots, taking care of elderly people, etc.
- About 15.6 million units of service robots for personal use will be sold between 2012 and 2015.
- About 93,800 new service robots for professional use will be installed for the period 2012-2015, e.g. defense applications (military systems), unmanned aerial vehicles, milking robots, medical robots, maintenance systems, rescue and security robots, etc.

As shown previously, robotic systems are gradually becoming an essential part of our everyday lives. Regarding to that, the most important external sensors which provide the robots with information about the surrounding environment are vision and force sensors. Most of the previous applications have implemented either vision or force sensor and many of them have used both. In some recent statistics, it can be found the following:

- In 2010, the total vision industry turnover increased by 34.8% (year-on-year). Moreover, in 2011 a further growth in sales of 20% is estimated for the European machine vision sector (EMVA, 2012), so it is noticeable that every year the using of vision sensor is growing rapidly.

- On the other hand, the Kinect camera which is the most recent vision system has claimed the Guinness World Record of being the "fastest selling consumer electronics device" after selling a total of 8 million units in its first 60 days. Furthermore, 24 million units of the Kinect sensor had been shipped as of January 2012 (Epstein & Zach, 2013), innumerable amount of them have been recently used in robot applications.

1.2 Problem formulation

In advanced robot applications, the challenge today is that the robot should perform different successive subtasks to achieve one or more complicated tasks similar to human. Hence, this kind of tasks required to combine different kind of sensors in order to get full information about the work environment. However, from the point of view of control, more sensors mean more possibilities for the structure of the control system. As shown previously, vision and force sensors are the most common external sensors in robot system. As a result, in scientific papers it can be found numerous control algorithms and different structures for vision/force robot control, e.g. shared, traded control etc. The lacks in integration of vision/force robot control could be summarized as follows:

- How to define which subspaces should be vision, position or force controlled?
- When the controller should switch from one control mode to another one?
- How to insure that the visual information could be reliably used?
- How to define the most appropriated vision/force control structure?

In many previous works, during performing a specified task one kind of vision/force control structure has been used which is pre-defined by the programmer. In addition to that, if the task is modified or changed, it would be much complicated for the user to describe the task and to define the most appropriated vision/force robot control especially if the user is inexperienced. Furthermore, vision and force sensors are used only as simple feedback (e.g. vision sensor is used usually as position estimator) or they are intended to avoid the obstacles. Accordingly, much useful information provided by the sensors which help the robot to perform the task autonomously is missed.

In our opinion, these lacks of defining the most appropriate vision/force robot control and the weakness in the utilization from all the information which could be provided by the sensors introduce important limits which prevent the robot to be versatile, autonomous, dependable and user-friendly. For this purpose, helping to increase autonomy, versatility, dependability and user-friendly in certain area of robotics which requires vision/force integration is the scope of this thesis. More concretely:

1. **Autonomy:** In the term of an automatic decision system which defines the most appropriated vision/force control modes for different kinds of tasks and chooses the best structure of vision/force control depending on the surrounding environments and a priori knowledge.

2. **Versatility:** By preparing some relevant scenarios for different situations, where both the visual servoing and force control are necessary and indispensable.
3. **Dependability:** In the term of the robot should depend on its own sensors more than on reprogramming and human intervention. In other words, how the robot system can use all the available information which could be provided by the vision and force sensors, not only for the target object but also for the features extraction of the whole scene.
4. **User-friendly:** By designing a high level description of the task, the object and the sensor configuration which is suitable also for inexperienced user.

If the previous properties are relatively achieved, the proposed robot system can:

- Perform different successive and complex tasks.
- Grasp/contact and track imprecisely placed objects with different poses.
- Decide automatically the most appropriate combination of vision/force feedback for every task and react immediately to the changes from one control cycle to another because of occurrence of some unforeseen events.
- Benefit from all the advantages of different vision/force control structures.
- Benefit from all the information provided by the sensors.
- Reduce the human intervention or reprogramming during the execution of the task.
- Facilitate the task description and entering of a priori-knowledge for the user, even if he/she is inexperienced.

Fig. 1.1 illustrates the main outline and the connection map between the chapters in this thesis. The number after every title will refer to chapter number. As shown in Fig. 1.1, the theoretical core of this work is the automatic decision system for fusing of vision and force information which is explained in chapter 4 in detail. The practical implementations are illustrated in chapter 5. This work presents some critical problems in three different applications and it illustrates how the integration of vision/force sensor can solve them and improve the robot performance. In section 5.1 in chapter 5, impact control has been improved during contact tasks. In section 5.2 in chapter 5, shelving and retrieval system which is situated for sorting objects in library milieu is presented. The last section 5.3 in chapter 5 has illustrated the task of handing-over objects from/to human hand. This chapter is supported with numerous experimental results and measurements for every task.

As shown in Fig. 1.1, for every practical task, an image processing algorithm has been proposed. The chapter 2 illustrates all the proposed image processing algorithms expect the adaptive algorithm for color detection (for contact task) which is explained in section 5.1.1.3 in chapter 5. The image processing algorithms in chapter 2 consist of three sections. In section 2.1 in chapter 2, the algorithm for detecting simple shape objects such as books is illustrated. This algorithm is fit for library scenario. Section 2.2 in chapter 2 presents the algorithm for detecting the complex shape object, this algorithm is implemented for detecting model-free objects located on flat surface (table, conveyor, etc.) before handing it to the human hand.

During object grasping from flat surface, new approach of visual servoing is proposed in chapter 3 (4x2D visual servoing) which benefits from the properties of Kinect camera. This approach helps the robot system to grasp object from flat surface even if the object is moving. The last section 2.3 in chapter 2 proposes two algorithms for detecting any object carried by human hand: 1. Skin color based approach, 2. Wrist model based approach. These algorithms help the robot system to detect and segment unknown object carried by human hand during handing-over task. All the proposed algorithms of image processing in chapter 2 are supported with real images taken during the performing of the task.

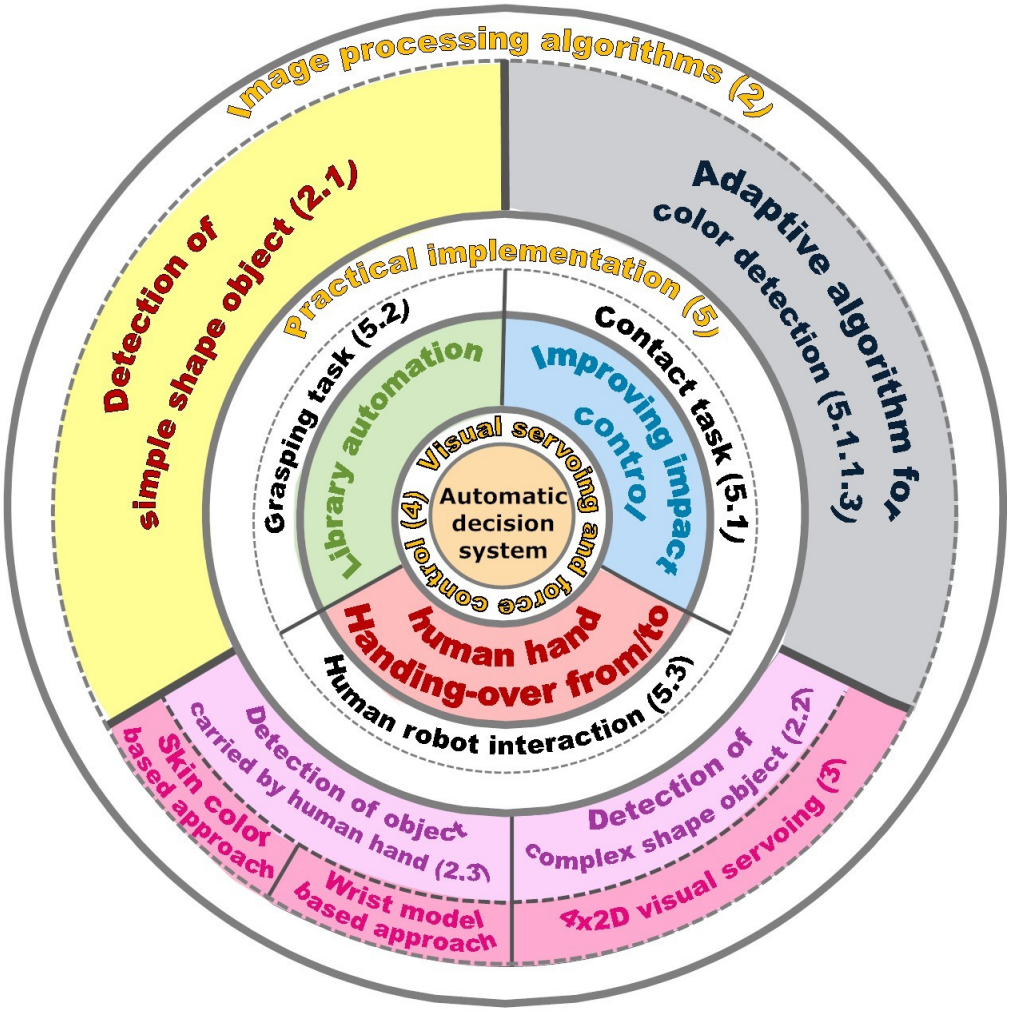

Fig. 1.1 Thesis outline

The most important previous works and developments which are related to every section in this thesis are reviewed and referenced starting from the past years and ending with the most recent literatures.

1.3 Theses

The main contributions of this thesis will be explained as follows:

1.3.1 Chapter 2 – Image processing algorithms

- Real time image processing algorithm for detecting different kinds of simple shape objects. This algorithm detects objects' position/orientation, characterizes and classifies them and then identifies the codes assigned to objects using SIFT features. The proposed vision algorithm has shown good performance for detecting the books boundary and for recognizing their labels without any limitations; if they are vertical, stuck together, inclined, etc. and even if the illumination or viewpoint have been changed.
- Real time image processing algorithm for detecting and segmenting complicated shape objects without any prior knowledge about their model in flat surface. The proposed algorithm is able to segment the objects even if they have bad contours, especially when the illumination is permanently changing or even if the objects are located on conveyor or movable surface.
- Fast real time image processing algorithm for detecting and segmenting any object carried by human hand. This algorithm contains two different approaches: 1. Skin color based approach, 2. Wrist model based approach. Skin color based approach takes opportunity to define exactly the contours of hand and object (even for objects with the same color as the human skin, if the human rotates his/her hand) and it doesn't depend on any model of the human hand or object. Skin color based approach uses RGB camera information as basic data and depth image as additional data. Because RGB data are less noisy than depth data the method shows good capability. Wrist model based approach can be implemented even if the human wears gloves or has Vitiligo disease. Furthermore, it is able to work in different light conditions starting from complete darkness and ending with different color-temperature lamps. In addition to that, it is able to segment the human hand and object even if they have the same color. The cycle time of both approaches is very short which gives us opportunity for real time visual tracking.

1.3.2 Chapter 3 – Visual servoing approach

- 4x2D visual servoing approach is special approach for Kinect camera which benefits from the images obtained by Kinect camera. The proposed approach combines the correspondent color and depth images to build two new images. Using these 4 images the control error signals will be calculated. In addition to that, this approach proposes a coordinate system which is called visible side coordinate system.

1.3.3 Chapter 4 – Automatic decision system

- Automatic decision system defines automatically the most appropriated vision/force control structures for different tasks depending on the surrounding environments and the preconditions of tasks, i.e. which subspaces should be vision controlled and which force controlled. This strategy will allow the robot to benefit from all the advantages of different vision/force control structures and to perform complex tasks with no need to be re-programmed or intervened by human. Furthermore, it proposes a simple interface for inexperienced user for entering the prior knowledge and for describing sensor configuration, task and the target object easily. Moreover it assumes a new coordinate system (approach) which is convenient to human conception. The automatic decision system illustrates how the robot system can use all the available information which could be provided by vision or force sensor such as graspability, user safety, etc. In other words, this work will not use the vision system as simple feedback or as desired position estimator but it will use it to extract the features of the whole scene and to understand the surrounding circumstances of the object during performing the task.

1.3.4 Chapter 5- practical implementation

- Improving the impact control during the switching from free space motion to constrained force control with the help of vision and force control.
- A new approach of robotic system which integrates vision and force feedback in order to shelve and retrieve imprecisely placed object according to their alphabetic/numeric codification intended for automating the libraries. This system can perform different grasping algorithms for books even if the target book has no neighbors, slop position or it is stuck between two other neighbors and there is no possibility to enter the parallel fingers of the robot around it.
- A new approach of assistant robot system for handing-over model-free objects from/to human hand. In the proposed system, the transfer object from/to human hand is performed exclusively by robot and the human has been considered as the weakest part in this task (elderly, blind or disabled). Furthermore, the fusion of vision and force will ensure the safety of the user during the physical human robot interaction, it will ensure the fulfillment of the grasping/releasing task and it will make the robot able to react to the motion of human hand during the interaction phase. The proposed system can tracks the human hand or the carried object starting from the free space motion and ending with the full physical human robot integration in real time.

1.4 State of the art

As shown previously, this work concerns on improving the integration of vision/force feedback and its implementation. As shown in Fig. 1.2, the state of the art could be divided into two sections: 1. Theoretical section which includes visual servoing, force control and vision/force integration. 2. Practical section which includes improving impact control, library automation and human robot interaction. The related work of the proposed image processing algorithms and the automatic decision system will be presented later in chapter 2 and chapter 4.

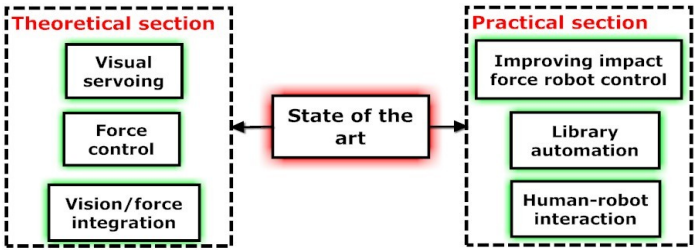

Fig. 1.2 Overview of state of the art

1.4.1 Visual servoing

The first who used vision feedback was Y. Shirai (Shirai & Inoue, 1973), he has described how a visual feedback loop can be used to correct the position of a robot to increase task accuracy. Visual servoing term has been first introduced by (Hill & Park, 1979). Since that time considerable efforts have been devoted to the visual control of robot manipulators, e.g. (Weiss, Sanderson, & Neuman, 1985) and (Weiss, Sanderson, & Neuman, 1987).

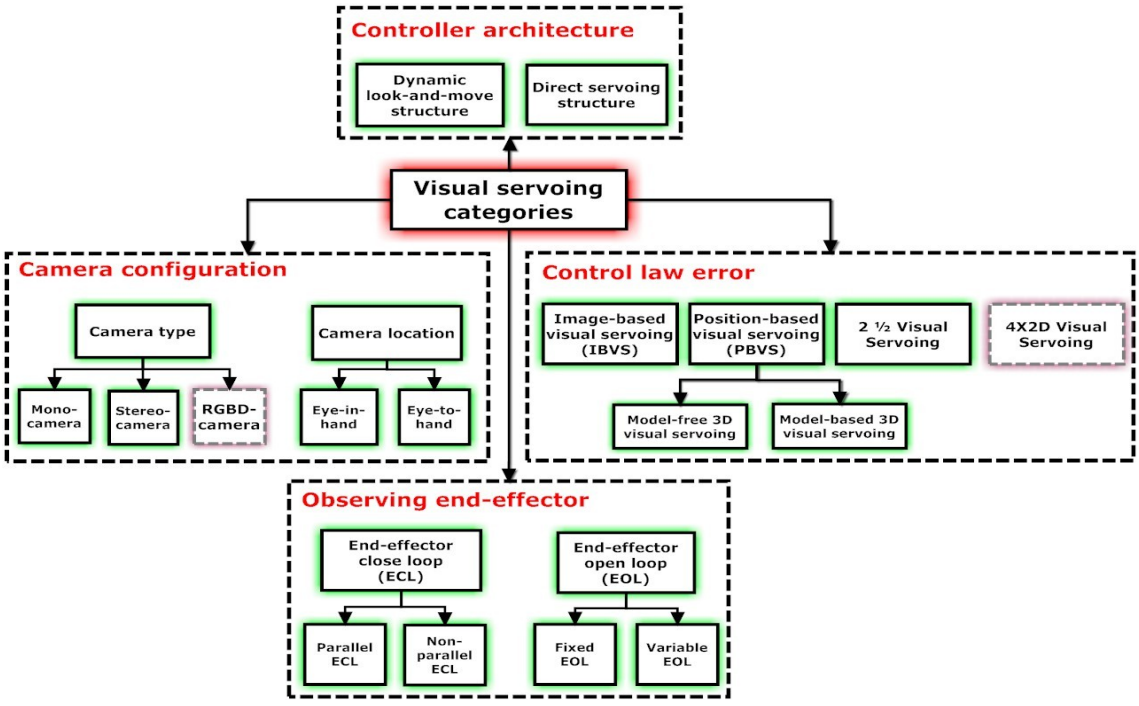

Fig. 1.3 Categories of visual servoing

Numerous papers, e.g. (Hutchinson, Hager, & Corke, 1996), (Hashimoto, 2003) and (Corke & Hutchinson, "Real-Time Vision, Tracking and Contorl", 2000), have discussed the principles of the arts of visual servoing for robot manipulators in details. Fig. 1.3 illustrates the main categories of the visual servoing approaches depending on different issues. These issues are: 1. Controller architecture, 2. Control law error, 3. Camera configuration, 4. Observing end-effector.

1.4.1.1 Controller architectures

Based on whether the control is applied to the joints directly or as a position command to a robot controller, visual servoing is divided into two types; 1. Direct servoing, 2. Dynamic look-and-move. Both configurations can be used in IBVS and PBVS approaches.

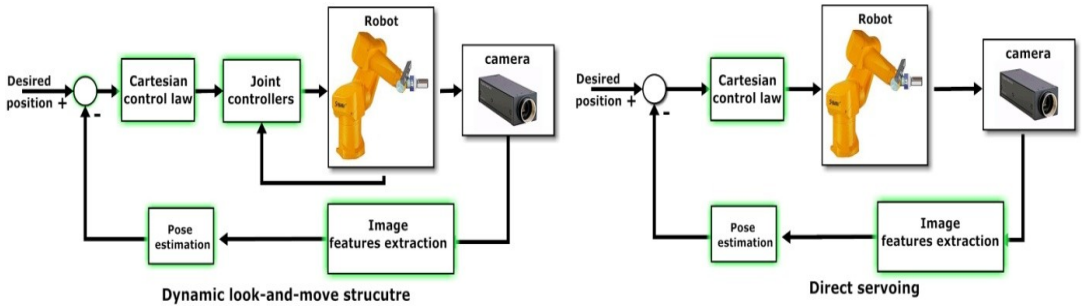

Fig. 1.4 Controller architectures (Weiss, Sanderson, & Neuman, 1987)

Dynamic look-and-move structure

Fig. 1.4 (left one) presents the dynamic look-and-move approach. The control architecture uses the vision system to provide set-point inputs to the joint-level controller, thus making use of joint feedback to internally stabilize the robot. A lot of image-guided systems have implemented the dynamic look and move approach, e.g. (Hutchinson, Hager, & Corke, 1996) and (Baeten & Schutter, 2004) for many reasons: Firstly, the relatively low sampling rates available from vision make direct control of a robot end-effector with complex, nonlinear dynamics an extremely challenging control problem. Using internal feedback with a high sampling rate generally presents the visual controller with idealized axis dynamics. Secondly, many robots already have an interface for accepting Cartesian velocity or incremental position commands. This simplifies the construction of the visual servo system, and also makes the methods more portable. Thirdly, look-and-move separates the kinematic singularities of the mechanism from the visual controller, allowing the robot to be considered as an ideal Cartesian motion device. Since many resolved rate controllers have specialized mechanisms for dealing with kinematic singularities, the system design is again greatly simplified.

Direct servoing structure

On the contrary, the direct visual servo eliminates the robot (joints) controller entirely replacing it with a visual servo controller that directly computes the joint inputs, as shown in Fig. 1.4 (right one). More specifically, the motor inputs of the robot system are driven directly

by the vision controller without the intervention of low level joint controllers, e.g. (Best, Molengraft, & Steinbuch, 2009) and (Bugarin & Kelly, 2010).

1.4.1.2 Control law error

The main types of visual servoing which depending on the error used to compute the control law are: position-based visual servo system, image-based visual servo system and 2D ½ visual servoing.

Position based visual servoing

In the position based control system, the error will be computed in the Cartesian space. In other words, the features are extracted from the image and they are used to estimate the pose of the target object with respect to the camera coordinate system. In general, position-based visual servoing techniques can be classified into two groups: model-based and model-free visual servoing.

Model-based 3D visual servoing: In this case, 3D model of the target object is available. The desired camera pose with respect to the object frame could be estimated from the error between the 3D model of the target object and the current image features. Fig. 1.5 presents scheme of model-based visual servoing. The extracted features of the object will be compared with its model in order to estimate the object's pose respect to the camera coordinate system.

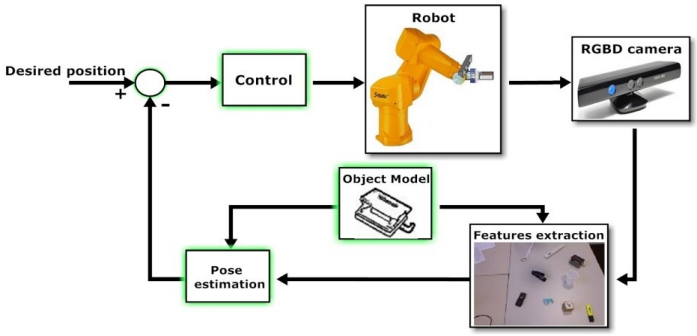

Fig. 1.5 Model-based 3D visual servoing

Model-free 3D visual servoing: In this mode, the 3D structure of the object or the environment is completely unknown. In other words, there is no 3D model of the target object (Model-free). Model-free 3D visual servoing is based on teaching-by-showing step. In the teaching phase the robot will be driven to the desired position in order to store the corresponding reference image features, after that when the object or the robot have been moved, the control error will drive the robot to the desired position from the current position using the displacement between the reference image features and current image features.

3D information could be reconstructed by fusing many 2D features, such as brightness, texture, color, size, and transparentness of the object. Furthermore photographic measurement or stereo camera could be used to reconstruct the 3D information. Photographic measurement uses the object shape and size to estimate the object position and orientation.

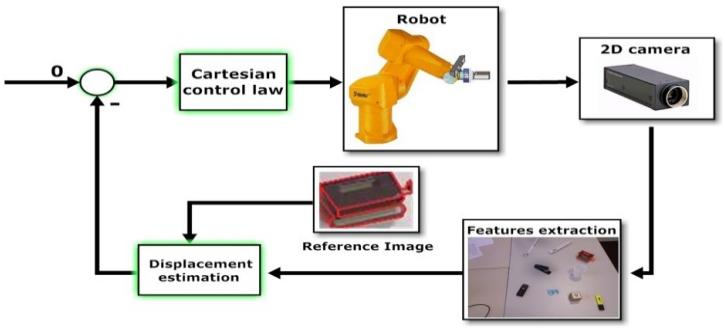

Fig. 1.6 Model-free 3D visual servoing

In general using projection equations we can calculate the corresponding points between the camera coordinate system (u, v) and the Cartesian coordinate system (x, y, z).

$$u = \frac{f}{z}\, x \tag{1.1}$$

$$v = \frac{f}{z}\, y \tag{1.2}$$

By assuming $\vec{X}_d$ is the desired position of the end-effector respect to the object and $\vec{X}_m$ is the measured position in Cartesian space corresponding to the current features in the image plane, we can write the control law:

$$\dot{\vec{\Theta}} = \lambda \cdot {J_a}^{-1} \cdot \left(\vec{X}_d - \vec{X}_m\right) \tag{1.3}$$

where ${J_a}^{-1}$ is the inverse Jacobian matrix and λ is a positive control gain. A comparison between model-based and model-free visual servoing could be found in (Hocaoglu, Bilen, Ozgur, & Unel, 2007), where reference (Malis, 2002) has proposed an unified approach to vision-based control (model-free and model-based) which can be used with a zooming camera whether the model of the object is known or not.

Image based visual servoing (2D visual servoing)

The image based visual servoing is a special case from model-free control because it does not need the knowledge of the 3D model of the target object and the control error will be formulated directly in 2D image space, so it will be called image error function.

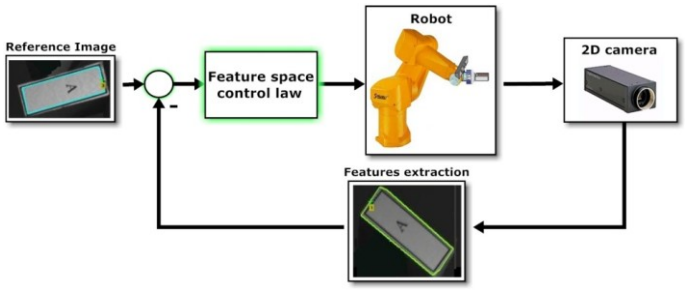

Fig. 1.7 Image based visual servoing

In the image-based visual servoing, the interaction matrix (image Jacobian matrix) will be used. This matrix describes how image feature parameters will change with respect to the change of the object or manipulator pose. This approach works with single camera and there is no need to the stereo computations.

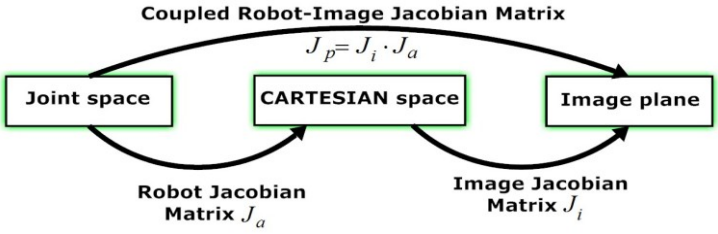

Fig. 1.8 Coupled robot-image interaction matrix

Let $\vec{s}_m$ be the current value of visual features observed by the camera and $\vec{s}_d$ be the desired value of these features to be reached in the image. By assuming camera-in-hand approach and the end-effector velocity is $v = [\vartheta^T \; w^T]^T$, the time variation of $\vec{s}_m$ related to the camera (end-effector) velocity will be calculated:

$$\dot{\vec{s}}_m = J_i \cdot \vec{v} \tag{1.4}$$

From joint space to Cartesian space we can write:

$$\vec{v} = J_a \cdot \dot{\vec{\Theta}} \tag{1.5}$$

By combining (1.4) and (1.5)

$$\dot{\vec{s}}_m = J_i \cdot J_a \cdot \dot{\vec{\Theta}} = J_p \cdot \dot{\vec{\Theta}} \tag{1.6}$$

The control law will be:

$$\dot{\vec{\Theta}} = \lambda \cdot {J_p}^{-1} \cdot (\vec{s}_d - \vec{s}_m) \tag{1.7}$$

where λ is a positive control gain.

Many papers have compared between IBVS and PBVS e.g. (Chaumette & Hutchinson, 2006) and (Chaumette & Hutchinson, 2007). In (Hafez, Cervera, & Jawahar, 2008) it has been proposed a hybrid visual servoing by boosting IBVS and PBVS.

In general, both methods position-based and image-based visual servoing have some drawbacks, e.g. in the position-based approach there is no performed control in the image space, which could imply that the target object may get out of the camera view during the servoing. Furthermore, a model of the target or teaching phase is needed to estimate the pose of the camera with respect to the target. In the image-based approach the depth estimation is needed to compute the interaction matrix and to design the control law. In addition to that, the matching between the desired and the current position will be insured only when the current position is in a neighborhood of the desired position.

A new approach was suggested by (Malis, Chaumette, & Boudet, 1999) in order to reduce these drawbacks. This approach is called 2 ½ D visual servoing.

2 ½ D Visual Servoing

2 ½ D visual servoing (Malis, Chaumette, & Boudet, 1999) is based on the estimation of the partial camera displacement from the current to the desired camera poses at each iteration of the control law. The extracted data from the partial camera displacement allow designing a decoupled control law which controls the six camera degree of freedom. In other words, 2 ½ D visual servoing can be used to decouple the translational and the rotational control loop. Since the position and orientation errors is controlled by two independent control loops, there is possibility to combine the advantages of IBVS and PBVS by selecting adequate visual features which part of them are defined in 2D and the other part in the 3D. Hence, the control scheme will always converge and avoid the singularities. Reference (Malis, 2002) and (Kragic & Christensen, 2002) have described the advantages and disadvantages of IBVS, PBVS and 2½ D approaches with comparison between them. Reference (Goncalves, Ferreira, & Pinto, 2002) has compared the behavior of 2D and 2½ D visual servoing in achieving a desired goal by planar robot. The overview of 2½ D visual servoing structure is illustrated in Fig. 1.9.

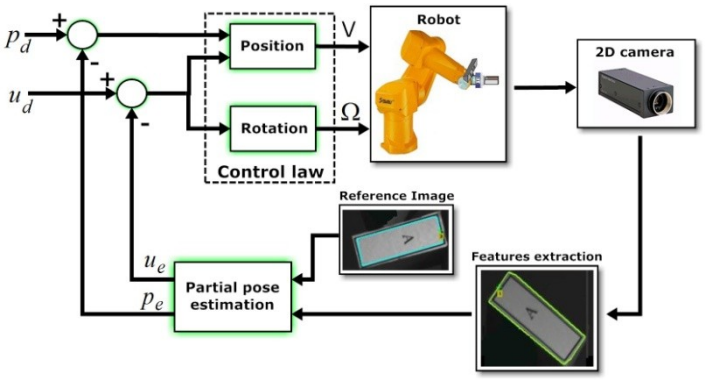

Fig. 1.9 2 ½ D visual servoing

where u_e and p_e are estimated rotation and position control vector, u_d and p_d are desired rotation and position control vector. In this thesis a new visual servoing approach will be introduced in chapter 3 which is called 4X2D visual servoing approach aimed to perform visual servoing using RGBD camera.

1.4.1.3 Camera configuration

Camera configuration includes two categories which are: 1. Camera type and 2. Camera location.

Camera type

In general, there are three main different cameras which are used for visual servoing: 1. Mono-camera; 2. Stereo-camera and recently 3. RGBD-camera.

- Mono-camera: for instance, 2D single camera is installed in a way that the optical axis is perpendicular to the workplace. The height between the camera and the environment should be known; in this case the three-dimensional feature position can be determined.
- Stereo-camera: consists of 2D single cameras monitoring the workplace. Both cameras should have known position, furthermore the characteristics of the images from both cameras should be together associated which is called correspondence problem (Yakimovsky & Cunningham, 1978). From the extracted features of both images (two-dimension projection of the scene) the vision system can reconstruct the depth information (Pari, Sebastian, Angel, & Rueda, 2009) and (Kase, Maru, A.Nishikawa, & Yamada, 1993). Depth information could be acquired using motion techniques which is similar to the stereo-camera expect that the scene will be captured by moving camera from different angles.
- RGBD camera: is novel sensing system which captures RGB images within every pixel the depth information is provided. An example of RGBD camera is Kinect camera which was originally intended to be a motion sensing input device for the Xbox 360.

Camera location

Visual servoing systems can typically be divided into two types depending on the camera location according to manipulator, as shown in Fig. 1.10: 1. Eye-To-Hand and 2. Eye-In-Hand. In the Eye-In-Hand configuration the camera has mounted on the robot's end-effector. In this case, the relationship between the position of the camera and the position of the end-effector is constant. This relationship is represented by ${}^{e}\mathrm{x}_{c}$. The position of the target relative to the camera frame is represented by ${}^{c}\mathrm{x}_{t}$, where the position of the target relative to the base coordinate system of the robot is represented by x_{t} , and the position of the robot's end-effector relative to the base coordinate system of the robot is represented by x_{e}.

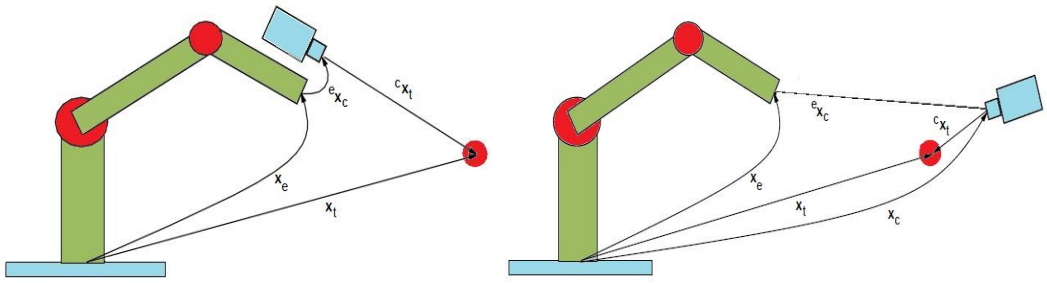

Fig. 1.10 Camera configuration (Eye-In-Hand and Eye-To-Hand)

In the other approach (Eye-To-Hand) the camera has fixed in the workspace. Hence, the camera is related to the base coordinate system of the robot by x_c and to the object by ${}^{c}x_{t}$. Camera calibration must be performed offline in order to determine the intrinsic camera parameters such as focal length, pixels pitch etc. At the same time, the position x_c, with respect to the world coordinate system should be determined.

1.4.1.4 Observing end-effector

Visual servo control can be also classified depending on whether the end-effector is observed or not into two types: 1. End-effector close loop and 2. End-effector open loop, as shown in Fig. 1.11.

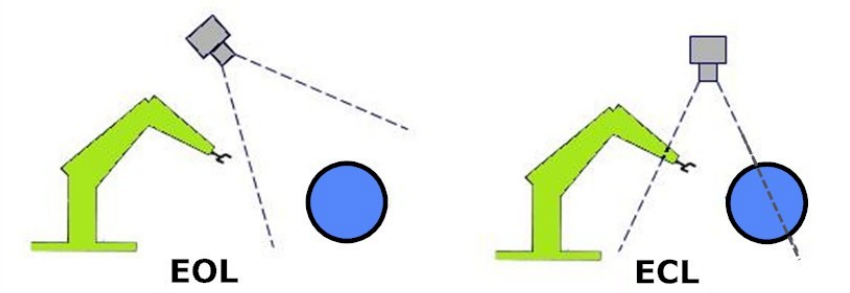

Fig. 1.11 EOL and ECL configuration

End-effector close loop (ECL)

The end-effector is being observed, An ECL setup may further be divided into two cases depending on the object constraint and the optimal camera positioning: If the image plane is taken parallel to the tool frame coordinate system, this setup will be called parallel configuration, otherwise non-parallel setup is chosen. The main advantage of an ECL configuration is that the tool always lies in the camera field of view. The main disadvantage may be the occurrence of occlusion of the object by the tool, and this making a straightforward vision measurement impossible (in parallel case).

End-effector open loop (EOL)

The end-effector is not seen by the vision system. In this approach also two subcases are possible: Either the relation between camera coordinate system and the tool coordinate system is fixed or it is variable. 1. Fixed EOL: the camera has a constant relative position from the tool coordinate system. 2. Variable EOL: the position of camera may change without actually changing in the tool position. The main advantage of an EOL configuration is the absence of occlusion of the object by the tool. Hence, easy image processing algorithms can be used. The main disadvantage is the needing of more complicated controller. If the image will be used in the tool coordinate system an extra control should be involved in the variable relation between the camera coordinate system and the tool coordinate system.

Reference (Chesi & Hashimoto, 2002) has compared the stability of the visual servoing in respect to the camera configuration eye-in-hand and eye-to-hand, where reference (Flandin, Chaumette, & Marchand, 2000) has presented a cooperation approach which integrates a fixed camera with one mounted on the robot.

1.4.2 Force Control

Robotic force control has been studied since the 1950's, when Goertz (Goertz, 1954) invented mechanical master-slave manipulators and then implemented these as electric-servo manipulators with force reflection. More details about force control history could be found in (Whitney, 1985). Force or torque is measured by measuring the strain induced by the force applied to an extensible element. Therefore, an indirect measurement of force is obtained by means of measurements of small displacements. Hence, the basic component of a force sensor is the strain gauge (Stefanescu, 2011) which uses the change of electric resistance of a wire under strain, i.e. senses the acting forces by measuring the deformation of the sensor. Currently, force sensor is one of the fundamental requirements for the success of a manipulation task during the interaction between manipulator and environment. In general six components (three forces and three torques as shown in equation (1.8)) could be measured by force sensor.

$$\vec{\mathrm{F}}^{\mathrm{T}} = \left[\mathrm{F_x}, \mathrm{F_y}, \mathrm{F_z},\right],\ \vec{\mathrm{M}}^{\mathrm{T}} = \left[\mathrm{M_x}, \mathrm{M_y}, \mathrm{M_z},\right] \tag{1.8}$$

Fig. 1.12 presents a manipulator holding object during the constraint motion. In this case, forces and torques will be generated in the contact point (contact coordinate system C) between the manipulator and the environment. Whereas, the force sensor will measure the applied forces/torques in the sensor coordinate system (where the force sensor is installed, in this case before the gripper). By assuming the forces/torques in the contact coordinate system are:

$$\vec{F}_C^T = \left[F_{Cx}, F_{Cy}, F_{Cz},\right],\ \vec{M}_C^T = \left[M_{Cx}, M_{Cy}, M_{Cz},\right] \tag{1.9}$$

The forces/torques in the sensor coordinate system are:

$$\vec{F}_S^T = \left[F_{Sx}, F_{Sy}, F_{Sz},\right],\ \vec{M}_S^T = \left[M_{Sx}, M_{Sy}, M_{Sz},\right] \tag{1.10}$$

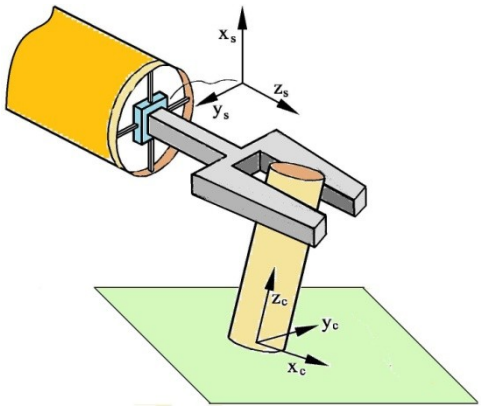

Fig. 1.12 Manipulator during constraint motion

Hence, a rotation matrix ${}^{C}R_S$ between contact coordinate system and sensor coordinate system will transform the measured values by the sensor to the actual forces/torques at the contact point as follows:

$$\vec{F}_C = {}^{C}R_S \cdot \vec{F}_S \tag{1.11}$$

For the torques, as is known there is relation between the forces and torques:

$$\vec{M} = \vec{r} \times \vec{F} \tag{1.12}$$

Hence,

$$\vec{M}_C = {}^{C}R_S \cdot \vec{M}_S + {}^{C}\vec{r}_S \times {}^{C}R_S \cdot \vec{F}_S \tag{1.13}$$

Where ${}^{C}\vec{r}_S$ is the vector between the sensor coordinate system and the contact coordinate system.

The main consideration when the robot contacting a surface is the controlling of the force amount which will be applied on the surface by the tool. Furthermore, ensuring the manipulator stability is required in force control, which could be difficult especially during initial contact between the tool and the surface. The goal of force control is to regulate the measured contact force ($\mathrm{F_m}$) to a constant desired force ($\mathrm{F_d}$) along the constraint surface as follows:

$$\mathrm{F_m} \rightarrow \mathrm{F_d}\,; \quad \mathrm{E_f} = (\,\mathrm{F_d} - \mathrm{F_m}\,) \rightarrow 0 \tag{1.14}$$

Some applications of force/torque sensor e.g. polishing, deburring, milling, cooperated handling and assembly operations are proposed in (Wang, Zhang, Zhang, & Fuhlbrigge, 2008), (Spiller & Verl, 2012) and (Lange, Suppa, & Hirzinger, 2012).

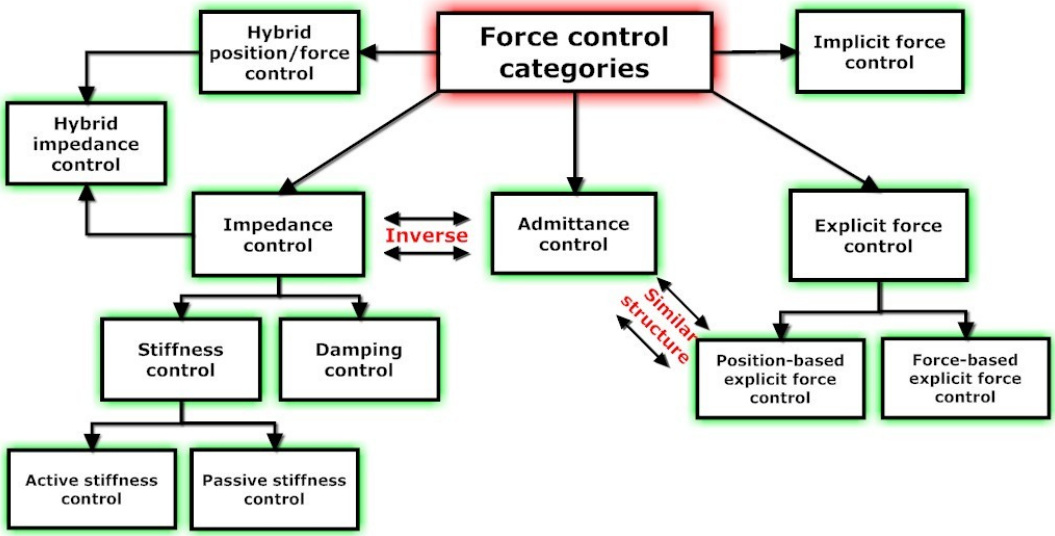

Fig. 1.13 Force control categories

Actually in scientific papers it can be found a huge number of force robot control algorithms with different structures. However, the main categories of force robot control are presented in Fig. 1.13 and they are explained in detail in (Whitney, 1985) and (Zeng & Hemami, 1997). In this section, an overview about these categories will be illustrated.

1.4.2.1 Explicit force control

In explicit force control, the measured force in the Cartesian space will be directly used in the loop feedback to calculate the control error. According to (Volpe & Khosla, 1993), robot explicit force control could be divided into two categories: 1. Force-based explicit force control and 2. Position-based explicit force control.

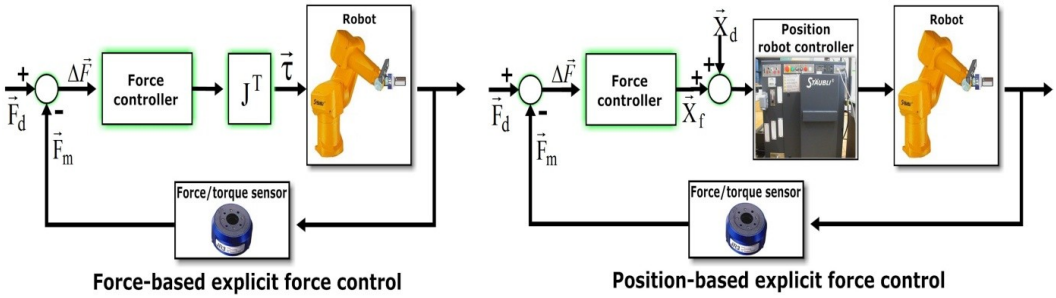

Fig. 1.14 Categories of explicit force control

Force-based explicit force control: The force controller of this approach compares the reference and the measured force values, processes them and then provides the robot system with the actuation signals as shown in Fig. 1.14 (left). The main problem of this approach, that it cannot be used in the commercial robot unless the robot controller has been replaced. Most commercial robots have built in position controllers and there is no direct access to control the actuator torques.

Position-based explicit force control: Probably position-based explicit force control has proposed mainly for the practical reasons, especially in the commercial robots. This approach consists of two parts: outer force loop and inner position loop. The outer force loop will provide a reference position to the inner position loop which will control the robot (Maples & Becker, 1986). The reference position will be calculated through the admittance (inverse of the impedance), therefore this approach is similar to the admittance force control approach.

1.4.2.2 Implicit force control

In implicit force control the force sensor is not involved (Zeng & Hemami, 1997). Instead, the position is controlled based on the predefinition of position for a desired force, i.e. the joint servo gains are adjusted to give the hand a particular stiffness matrix.

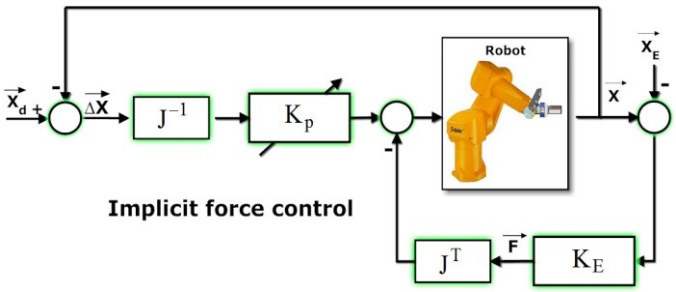

Fig. 1.15 Implicit force control

Fig. 1.15 presents the structure of implicit force control, where K_P is position feedback gain, K_E represents the stiffness matrix of the environment and the force/torque sensor together and $\vec{X}_E$ is the position of the environment during the contact as shown in Fig. 1.16 and equation (1.15).

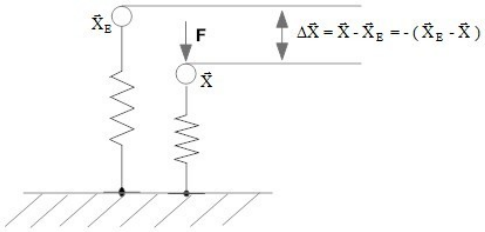

Fig. 1.16 Relation between force and position

$$\vec{F} = -K_E(\vec{X} - \vec{X}_E) \tag{1.15}$$

In (Osypiuk, Kröger, Finkemeyer, & Wahl, 2006) an improved implicit force and position control proposed based on a simple linear model.

1.4.2.3 Impedance control

Impedance control is a method to regulate the mechanical impedance of an end-effector of a robot manipulator in a desired value according to a given task. In (Hogan, 1985), Hogan has proposed the impedance control, where the manipulator control system should be designed not only to track the trajectory, but rather to regulate the mechanical impedance of the manipulator. The mechanical impedance Z defines the relation between the velocity $\dot{X}$ and the applied force F as follows:

$$Z = \frac{F}{\dot{X}} \tag{1.16}$$

As is known:

$$m\,\ddot{X} + k\,\dot{X} + c\,X = F \tag{1.17}$$

where m, k and m represent mass, damping and stiffness respectively. By using Laplace transformer:

$$\left(m\,S + k + \frac{c}{S}\right) S\,X(S) = F(S) \tag{1.18}$$

Then the mechanical impedance is:

$$Z(S) = \frac{F(S)}{S\,X(S)} = m\,S + k + \frac{c}{S} \tag{1.19}$$

As previously mentioned, Hogan has invented the method to control the end-effector impedance of a manipulator based on the measured position, velocity and force of the end-effector. Since then, many studies e.g. (McCormick & Schwartz, 1993), (Craig, 1989) and (Morel, Malis, & Boudet, 1998) have implemented the impedance control in various forms

depending on how the measured signals are used. The impedance control is one of the most important frameworks to control the interaction between the manipulators and the environment. As special cases of impedance control have proposed e.g. stiffness control (Salisbury, 1980) or damping control (Whitney, 1985).

1.4.2.4 Admittance control

The mechanical admittance A is the inverse of the mechanical impedance and it can be defined as follow:

$$A = \frac{1}{Z} = \frac{\dot{X}}{F} \tag{1.20}$$

Admittance control is similar to the position-based explicit force control and it has presented in (Whitney, 1985) and (Seraji, 1994).

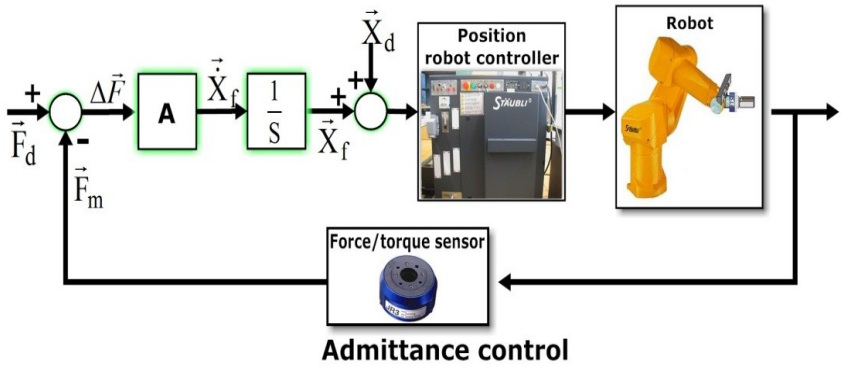

Fig. 1.17 Admittance control

Fig. 1.17 presents the structure of the admittance control which makes the necessary modifications of the system admittance in order to achieve the desired task. By comparing admittance and impedance control, admittance control focuses more on tracking the desired force (Zeng & Hemami, 1997).

1.4.2.5 Hybrid position/force control

There are two extreme modes of operation for a manipulator: 1. Position controlled motion (pure position) through free space, this operations process without contacting the environment, some industrial applications for this scheme are spray painting and pick-and-place tasks. 2. Force controlled motion (pure force) constrained by the environment, for example in milling, cutting, drilling and assembly. In most industrial applications manipulator must change from one mode to the other readily. However, most industrial robot applications need to combine position and force control, i.e. some directions are force controlled and the others are position controlled. Reference (Mason T. , 1981) has introduced a theory which has distinguished between natural and artificial constraints. Hybrid position/force control was proposed in (Raibert & Craig, 1981) depending on Mason theory.

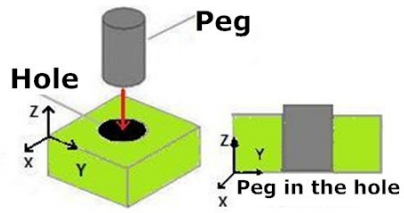

Fig. 1.18 Peg in the hole

Fig. 1.18 presents a task (peg in the hole) which needs to combine position and force control together. According to Mason theory (Mason T. , 1981), natural constrains describes the concrete contact situation for the velocities and forces/torques in the senses that speeds, torques and forces are impossible. In Fig. 1.18, it can be easily figured out that the six natural compliances are:

$$\begin{aligned} V_X = 0 \quad \omega_X = 0 \\ V_Y = 0 \quad \omega_Y = 0 \\ F_Z = 0 \quad M_Z = 0 \end{aligned} \tag{1.21}$$

The movements in the direction x and y which described in the equation (1.21) are not possible. In this case is an idealization is used, that the friction is neglected, therefore it is impossible to exert force in F_Z direction or torque in M_Z direction. Whereas, the artificial constrains determines which movements and forces/Torques in the given direction are possible. Logically to know that where it is not possible to perform the movements, it is certainly possible to exert forces order torques and vice versa.

$$\begin{aligned} F_X = 0 \quad M_X = 0 \\ F_Y = 0 \quad M_Y = 0 \\ V_Z = K_1 \quad \omega_Z = K_2 \end{aligned} \tag{1.22}$$

Whereas: K_1 the desired velocity in z direction and K_2 desired angular velocity in z direction.

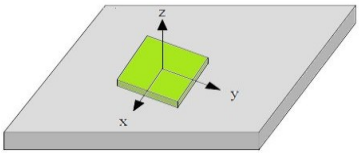

Fig. 1.19 Object on a table

Fig. 1.19 presents a task for the planar robot. In this case, the natural constraints will be as follows:

$$\begin{aligned} F_X = 0 \quad \omega_X = 0 \\ F_Y = 0 \quad \omega_Y = 0 \\ V_Z = 0 \quad M_Z = 0 \end{aligned} \tag{1.23}$$

Whereas, the artificial constraints are:

$$\begin{aligned} V_X = K_1(t) \quad M_X = 0 \\ V_Y = K_2(t) \quad M_Y = 0 \\ F_Z = 0 \quad \omega_Z = K_3 \end{aligned} \tag{1.24}$$

Fig. 1.20 illustrates the structure of hybrid position/force control. The matrix S determines the subspaces which are position controlled, i.e. the subspaces which are related to the natural constraints. In the same way, the complementary of matrix S is matrix $I - S$ which will be related to the artificial constraints and it will defines the force controlled subspaces.

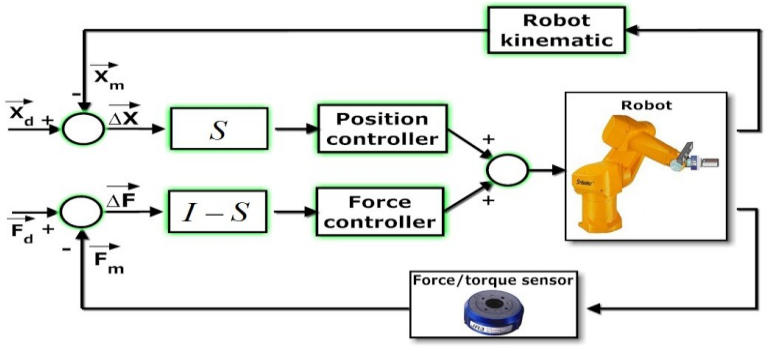

Fig. 1.20 Hybrid position/force control (Raibert & Craig, 1981)

S and $I - S$ are (6x6) binary matrixes (as shown in equation (1.25)). If S_i = 0 the i^{th} DOF will be forced controlled, otherwise it will be position controlled.

$$I - S = \begin{pmatrix} I - S_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & I - S_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & I - S_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & I - S_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & I - S_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & I - S_6 \end{pmatrix} \quad S = \begin{pmatrix} S_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & S_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & S_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & S_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & S_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & S_6 \end{pmatrix} \tag{1.25}$$

An improved structure has proposed in (Anderson & Spong, 1987) which combines the hybrid control and the impedance control in one control structure.

1.4.3 Vision/force integration

Vision provides global information on surrounding environment for motion planning and obstacle avoidance, whereas force allows adjusting robot motion so that local constraints imposed by environment are satisfied. In general visual feedback is used in gross motion and the force feedback is used in fine motion. When robot is far from an object the visual servoing is adopted and the relative position of the robot with respect to the object is calculated using vision sensor. When the robot is in contact with object any kind of interaction control strategy can be adopted (impedance control, force/position control), and the relative position of the robot with respect to the object is generated recursively using vision, force and/or joint position measurements. One of the first papers to realize the benefits of combining visual and force feedback is by (Shirai & Inoue, 1973). In (Nelson, Morrow, & Khosla, 1996), it is presented the combination of vision and force sensing within the feedback loop of a robot manipulator in three approaches: traded, hybrid and shared control.

As shown in Fig. 1.21, we have divided vision/force robot control into six types: two extreme cases (pure position control and pure force control), three further common approaches (traded, shared and hybrid control) and one approach proposed here (vision/guarded-force control). Traded, shared and vision/guarded-force control could be applied in every direction separately. However, hybrid control could be applied in multi directions.

- In pure position control all directions and orientations are position controlled. This representation could be used in free space or when the target position is previously known.
- In pure force control the motion is constrained by the environment, object or during physical interaction with the human.
- Traded control could be applied in every direction separately. The task-space direction is alternatively controlled using a vision or force sensor, i.e. manipulator motion is first controlled by visual feedback and then the controller switches to force control when the robot is near sufficient to the environment.
- In shared control, both vision and force sensors control the same direction of the task space simultaneously.
- Hybrid control allows visual servoing only in those directions that are orthogonal to directions along which force feedback is used. In other words, in hybrid control the vision and force control are simultaneously applied on separate directions, which is extension of hybrid position/force control.
- In vision/guarded-force control, the vision feedback will take the main role of the control loop, which means that the subspace will be approached under vision control. In other hand, the force sensor should be monitored, i.e. if the measured force value exceeds a specified threshold, the motion will be immediately stopped. A previous work (Gourishankar, Trybus, Rink, & Steil, 1987) has proposed the guarded force control but it was not integrated with the vision system.

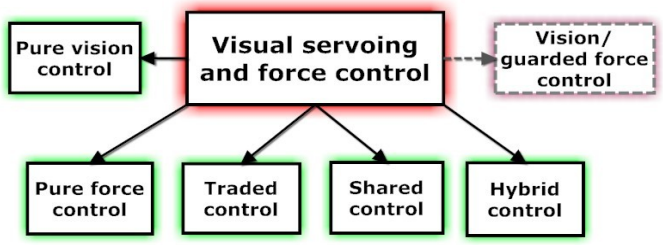

Fig. 1.21 Categories of visual servoing and force control

In general every control mode of vision/force feedback combination has advantages and disadvantages.

- The main benefit of traded control is the stable impact with the target surface can be achieved under force control without inducing a large contact forces. Whereas, using only force control the manipulator can easily become unstable at the first contact

moment, unless the force gains have extremely low values. The limitation of traded control is that the switching time from vision feedback to force feedback should be carefully defined in order to balance between the motion speed and the impact force.

- Shared control is useful on the surfaces which cannot be reliably detected with vision feedback and it is necessary at the same time that the manipulator reacts compliantly to the environments along the visual servoing direction. Its disadvantage appears when the vision system commands motions. The resulting accelerations causing oscillations of force control system because of end-effector inertial effects.
- Hybrid control is a little limited because it ignores much of the information provided by visual feedback because hybrid control uses vision control only the directions which are orthogonal to directions where the other types of feedback is used.
- Vision/guarded-force control is useful in the tasks where the vision feedback cannot reliably detect the target direction and there is no desired force will be applied on this direction. The main limitation of this strategy is that high contact forces could be occurred unless the effective mass of the manipulator is low or the threshold value is low tuned so that the end-effector can be quickly stopped before a large contact forces increase significantly.

Numerous papers have concerned on improving the integration of vision and force sensors, e.g. (Hosoda, Igarashi, & Asada, 1996), (Baeten, Verdonck, Bruyninckx, & Schutter, 2000) and (Baeten & Schutter, 2002). The general characteristics of these papers are the following:

- If robot is far from the object, the visual servoing is adopted and the relative position of the robot with respect to the object is calculated with the help of vision system.
- If robot is in contact with the object or with environment one kind of interaction control strategy is adopted, and the relative position of the robot with respect to the object is generated recursively using vision, force and/or joint position measurements.
- The specified control structure is used depending on the task and its circumstances.
- The main role of vision system is only to know the relative position of the robot with respect to the object.

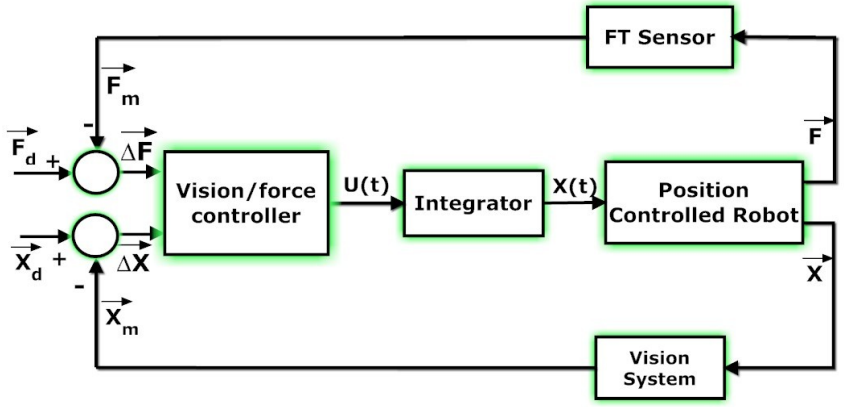

Fig. 1.22 Hybrid vision/force control

Fig. 1.22 shows the typical scheme of hybrid vision force control. Depending on the values of the controller and when the vision or force loop is activated, the category of the vision/force control will be defined. In this thesis, the proposed vision/force algorithms and the improved structure of vision/force robot control will be deeply illustrated in chapter 4.

1.4.4 Improving impact force robot control

Many automated manufacturing processes require robots to interact with environment and to perform force/torque interaction such as mechanical assembly. Importantly, impact forces occur while robot and environment are in contact. Robot manipulators and control systems can experience instability or poor control performance after impacting with an environment. Impact force may induce large interaction force which could harmful the robot manipulator or even the environment. Therefore, impact force problems have caught the attention of many researches and works. For instance, reference (Parker & Paul, 1988) has proposed algorithms and apparatus in order to control the impact force when the robot is rapidly grasping an object, reference (Weng & Young, 1996) has proposed a control scheme which deal with unexpected impacts when the robot interact with the environment which is inspired by human reflex. Another work (Lee, 1999) has developed approach which combines natural admittance control (NAC) with time delay control (TDC) for interaction control with unknown dynamics.

In (Lee, Park, Schrader, & Chang, 2003), nonlinear bang-bang impact controller is developed by using robust hybrid impedance/time-delay control algorithm. With the help of the developed controller the robot can successfully achieve contact tasks without changing control algorithms or controller gains during switching from free-space mode to constraint motion. More recently work (Ranko, Angel, Pedro, & Angel, 2006) has proposed architecture for force and impact control which controls the robot starting from free-motion until the contact with the environment.

All the previously mentioned works have tried to improve the impact force controller. In our opinion, for any robots system operating in various environments it is sometimes insufficient to use only one kind of feedback. In order to get full information about the work environment it is preferable to use different kinds of sensors. Vision sensors are among the best kinds of sensors which provide information over a relatively large area of the workspace without requiring contact with the environment.

Reference (Nelson, Morrow, & Khosla, 1995) has proposed different strategies which combine vision and force within the feedback loop of the robot. Furthermore, it has illustrated that the using of visual servoing will simplifies the force control problem. In this thesis, the vision and force feedback will combined in order to reduce the impact forces and to increase the performance of robot, by calculating the distance between robot's end-effector and the environment and reducing the speed according to it. The proposed algorithm of improving impact control will be described in chapter 5.1.

1.4.5 Library automation

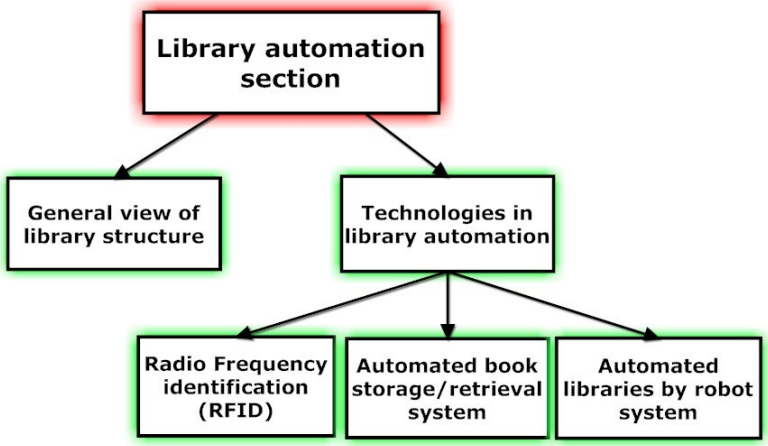

Fig. 1.23 Overview of library automation section

As shown in Fig. 1.23, this section will explain the main structure of the library. After that the technologies of the automated library will be presented briefly.

1.4.5.1 General view of libraries structure

In general, libraries are huge collection of information, books, sources, resources and other documents on various storage media such as audio tapes, videotapes, cassettes, CDs and DVDs. All these large amounts of materials should be arranged, generally each library has organized their materials according to classification system which represented by letters and numbers, the library workers include librarians and other professionals, paraprofessionals, clerical and technical personal. Usually, the tasks performed in the library are:

- Circulation (access services): Handles user accounts and the tasks of loaning, returning and shelving of materials
- Collection, development, order materials and maintain materials budgets.
- Technical services: catalogs and processes the new materials and cancels the deleted materials
- Stacks Maintenance: Re-shelves materials that have been returned to the library after customer use and shelves materials that have been processed by Technical Services. Stacks Maintenance also shelf reads the material in the stacks to ensure that it is in the correct library classification order

Now in the most libraries Self-check in/out machine Fig. 1.24 are available, the self-check provides users with a quick and easy way to check out materials otherwise the customer has to wait in line at the service desk. It also has the great benefit of allowing users to check out materials when the service desk is closed.

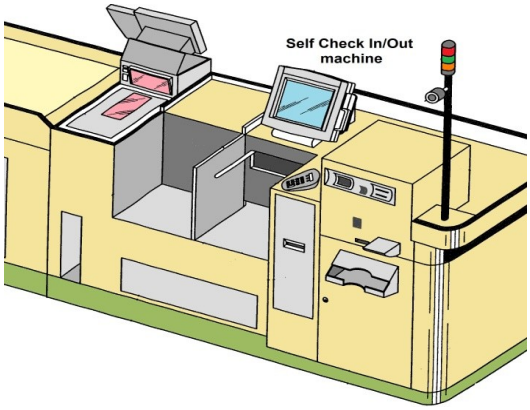

Fig. 1.24 Self check in/out machine

According to their conveyor system, the self check-in machine is sorted into two types; 1. In simple conveyor, the materials are transferred from check-in point to the first store stage. Here, one or more employers should sort the materials according to the types (science, romance, history and etc.) and put them inside the book trucks. After that other employers move these book trucks to the floor and department where the book should be shelved. 2. In more complicated conveyor, the system consists of a wide range of overlapped conveyors which cover the whole library (also between floors). This system sorts and transfers the materials to their department. In every department, one or more employers should re-shelve the books. This kind of libraries is half automated because the human is still needed in shelving the materials. Furthermore, the employers should read, analyze the codes of materials and then organize them according the classification code during the shelving tasks.

1.4.5.2 Technologies in library automation

Recently there are three technologies are used in the most recent automated library:

Radio Frequency identification (RFID)

RFID technology (Dhanalakshmi & Mamatha, 2009) helps to automate business processes and allows identification of large number of tagged objects like books, using radio waves. In this way, the information of the book and the library member are provided to the library management system and there is no need to manual typing. It also provides continuously monitoring of books movement across the gates using monitoring module. It can also search for books using searching module and RFID handheld reader. RFID technology speeds up the self check in/out processes, controls the theft and eases the inventory control in the library. Even though this method speeds up the searching tasks for a book, but the skilled workers are still needed in shelving and rearranging the returned materials and retrieving them again.

Automated book storage/retrieval system

These kinds of researches and inventions which handle with library automation were concerned on removing the requested books from their prescribed locations in the bookracks

and returning them. As shown in Fig. 1.25, one of the first automated storage/retrieval systems (Yoshie, 1997) is intended to automate the following operations: retrieving a book which requested by user from a stack room; conveying the book to a circulation counter; reposting the book again; the books are stored in containers and the containers are filed into or removed from stack room, when removal of a book is requested, a container storing the requested book is automatically removed from the stack room by crane, and then the container is conveyed to the circulation counter using conveyor system, then the employer searches in the container for the desired book and hands it to the customer.

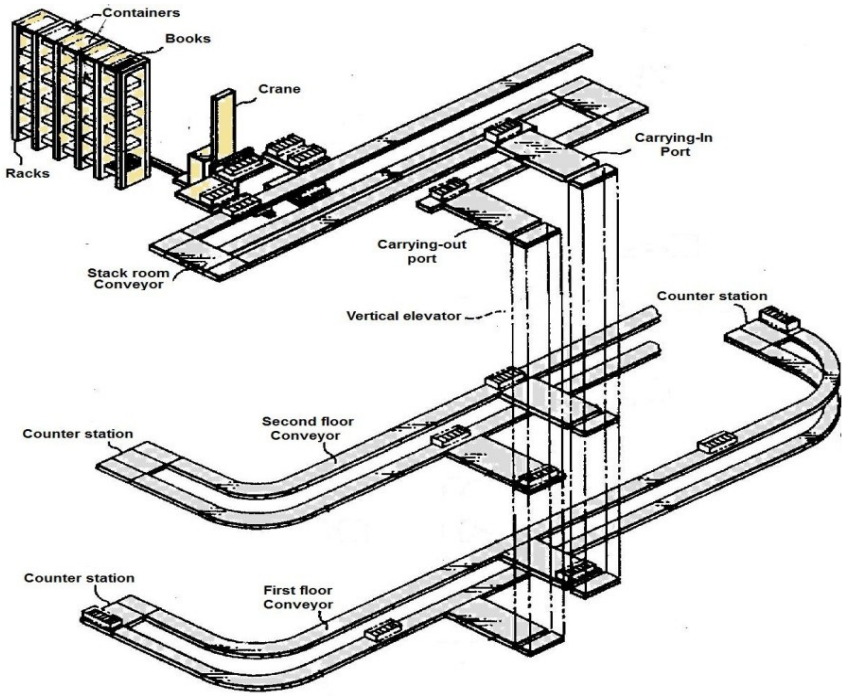

Fig. 1.25 Automated book storage/retrieval system (Yoshie, 1997)

When shelving of a book is requested, a container having an available storage space is automatically removed from the stack room and is conveyed to the circulation counter. The employer puts the books in the container and returns the container automatically to the stack room. When the container is brought into the stack room, the bar codes of all stored books in the container are read to update the stored contents indicating the locations of the books. In this system the containers should be consisted of one row of the books in order to ease the code reading tasks.

To improve the storage efficiency of the stack room a new mechanism of container, as shown in Fig. 1.26, are developed so each container consists of two or three rows and the code of all stored books in the container can be easily read. Each container consists of nine sub-containers from A to I, when a sub-container including a book requested to be removed is raised to a position higher than that of the remaining sub-containers.

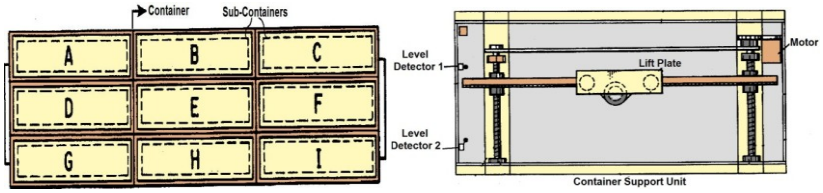

Fig. 1.26 Container structure (Yoshie, 1997)

It is clear that both systems have many disadvantages, such as; in every requested or filed order the system transports one complete container even though if the user or the employer needs only one book. Hence, the system will bear additional encumbrances which: can be dismissed, waste the energy and need more maintenance. The bar codes of all stored books in the container should be read when the container is brought into the stack room, in order to update the location of the books even though if only one book in the container is changed. This means waste of time and in some cases the location of the book will not be known until the container is brought back to the stack room.

Automated libraries by robot systems

More recently patents and researches, as is shown in Fig. 1.27 and Fig. 1.28, have proposed a robot system to automate the libraries; the first suggested system (Timothy & D.Plutt, 2003) contains multiple independent robots for concurrently manipulating multiple tasks.

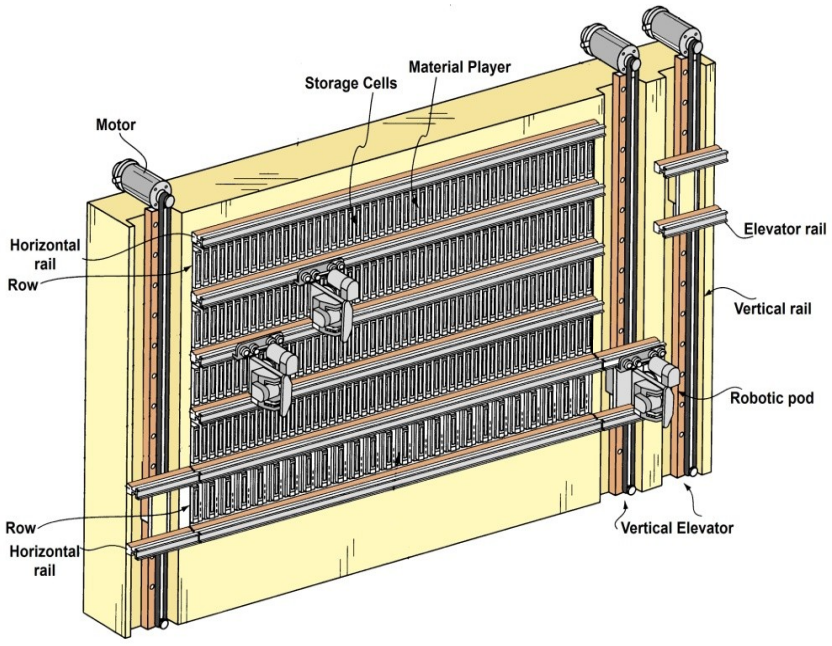

Fig. 1.27 multiple independent robots structure (Timothy & D.Plutt, 2003)

The library system comprises a two-dimensional array that contains cells which storage the materials, and a system of rails is used to guide the robotic pods through all the locations

(horizontal and vertical movements) of the array, the robotic pods contain a moveable carriage which transports robotic components, materials pickers and bar code reading devices.

The second recently patent (Nakano, Y.Kihara, Sakimoto, & Hayashi, 2003), as shown in Fig. 1.28, has suggested a mobile robot system for book retrieving and reposting. The robot has a carriage that moves itself around a plurality of shelf sections. Mounted on the carriage are a hand mechanism, controller for controlling three-dimensional motion of the hand mechanism and a computer for determining both the book identification information for a requested book and the shelf section storage location information of the requested book.

The robot's hand mechanism includes an identification information recognition unit for detecting shelf section identification information indicated on the bookrack, as well as book identification information; a book pullout mechanism for pulling-out a requested book from a row of books in a shelf section; and a book-holding mechanism for grasping the pulled-out book and removing it from the bookrack.

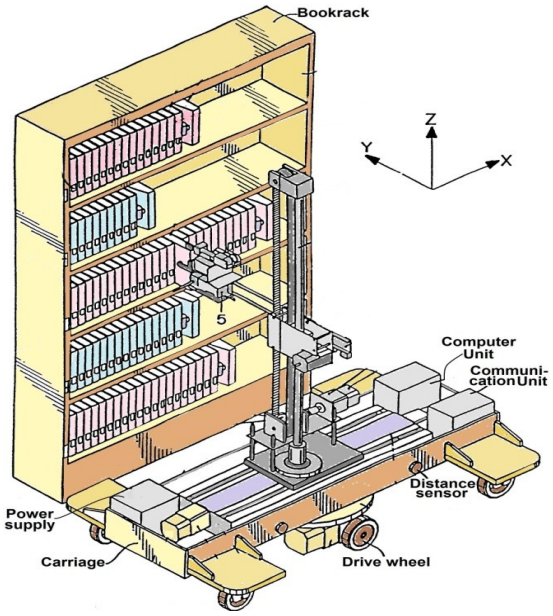

Fig. 1.28 Automated library with mobile robot (Nakano, Y.Kihara, Sakimoto, & Hayashi, 2003)

Another project which is called Comprehensive Access to Printed Materials (CAMP) has been developed in Johns Hopkins university (Suthakorn, Lee, Zhou, Thomas, & Choudhury, 2002). This project has proposed a robot which is able to extract and to browse books from the library. This project has focused on robot design and control systems. Furthermore, it has assumed that the books have the same sizes which are stored in special cases one by one with constant space between them on a well structured and specially designed library.

The disadvantages of the previously mentioned work summarized as follows: 1. Many additional hardware equipments on the previously systems could be eliminated by using the vision system, such as identification information recognition unit, huge memory unit for

storing the book location, Wedge-shaped bookend is placed at the end of the row of books and rocking guide rollers which are used to facilitate the re-shelving of books by the robot and guiding path; which save money, maintaining and fault detection time . 2. All the locations of the shelf section are stored in the library, such as bookrack and book locations should be prescribed in details; which required huge capacities of memories, and when any changing in library structure happens, the system should be updated every time. 3. If one book is in wrong location placed, for unknown reason, there is no possibility to find this book again using the previously systems. Hence, using vision system in this kind of tasks will save the money, time of fault detection and maintenance operations; it will make the system more adaptively to handle with different types of workplaces and it will improve the performance of the system without needing previous information about the environment.

Three other projects, which are implemented for similar goals in library automation and library service, have confirmed the importance of using the vision system. In the first one (Tomizawa, Ohya, & Yuta, 2003), they have focused their efforts in improving the mechanism of book browsing. The main goal of this project is to help human to browse books located in a library from a remote location via internet. However, the section of extraction and return the book from the bookshelf is simply addressed.

Another work which is recently implemented (Heyer, Enjarini, Fragkopoulos, & Graeser, 2012) has proposed a service robot which is able to work in library to help the disabled people using vision system. This work has assumed that the robot can grasp only the most right book from upper shelf. Furthermore, it has supposed that every two neighbor books should have different height and different depth as shown in Fig. 1.29, which makes the segmentation by disparity map easier and it makes the grasping algorithms simpler.

Fig. 1.29 Proposed books

The third project (Prats, Sanz, & Pobil, 2005) is an interesting work which has similar goals of ours represented by improving the grasping and extracting the books from their shelves using vision/force robot control. This work has confirmed the importance not only of the vision sensor but moreover the combining of vision and force feedback in order to solve the problems of the extracting the book from the shelf. However, this project has proposed a simple hybrid vision/force control in two degrees of freedom with the help of a parallel jaw gripper used to accomplish the task of extracting a book from the shelf and it is endowed with special purpose fingertips. This project has assumed the following: 1. The books are not pressed together on the shelf in such a way as to impede the insertion of the gripper fingers. 2. The hybrid vision/force control has been implemented only in two degrees of freedom: In the first one, the image-based visual servoing has been used in the parallel direction to the book. Whereas, the second one is the perpendicular direction to the book which is guided by force sensor in order to establish the contact with the book. However, the proposed contributions and improvements of this thesis will be deeply illustrated in chapter 5.2.

1.4.6 Human robot interaction

Human robot interaction is wide term which consists of different oriented subjects e.g. human robot interaction depending on human motion primitives and human instructions such as (Cuntoor, Collins, & Hoogs, 2012), (Mangin & Oudeyer, 2012); or humanoid robot intended to work in human environment such as (Sian, Sakaguchi, & Yokoi, 2006) and (Kemp, Edsinger, & Torres-Jara, 2007). One of main topics in human robot interaction is the physical interaction which concerns to improve the handing-over tasks between human and robot.

Various researches have suggested different approaches of service robot systems which are able to interact with the human, e.g. in (Diftler, Ambrose, Tyree, Goza, & Huber, 2004) a mobile autonomous humanoid robot is proposed. The proposed system assists human at the Johnson Space Center with tool handling tasks. Robonaut is the first humanoid built for space with 43 degrees of freedom and it is supported with stereo vision system. The stereo-based vision system capitalizes on object shape to track the pose (position and orientation) of well-defined objects, such as wrenches and tables. In other words, it matches the real image with large sets of 2D templates for acquisition, and 3D templates for pose estimation. Hence, a priori knowledge about the characteristics and the model of the object are needed.

In (Bischoff & Graefe, 2004), a large number of functionalities were integrated in the humanoid robot HERMES to interact and to communicate with human. HERMES robot runs on 4 wheels and it is supported with two articulated arms with 6 degrees of freedom. Each arm is equipped with a two-finger gripper to carry a payload of 2.0kg. Two video cameras mounted on independent tilt drive units in order to control the head platform. A radio Ethernet interface allows communicating via a LAN or the Internet. HERMES robot is able to communicate with people in different language, to localize itself, to build maps and to manipulate various objects.

The previous mentioned works have designed a kind of humanoid robots which are able to interact and to communicate with human or even to work in human environment. However, they didn't concentrate on improving the handing-over tasks between human and robot. In the most service robots application, the human will not interact directly with the robot, but a transported object will serve as a connection bridge between them. In other words, the system will transfer objects between the human hand and the robot hand.

Numerous papers have focused only on the problem of handing-over between human and robot, e.g. in (Edsinger & Kemp, 2007) experimental results are presented which demonstrate handing objects to robot and taking objects from it in response to reaching gestures. In this work the human solves a potentially difficult grasping problem for the robot by directly placing the object within the robot's hand in a favorable configuration. Another work (Cakmak, Srinivasa, Lee, Forlizzi, & Kiesler, 2011) has focused on handing over objects to humans depending on human preferences, which means that the robot will hand over the object with a planned configuration using a kinematic model of a human and learnt from examples given by other humans. Reference (Huber, Rickert, Knoll, Brandt, & Glasuer, 2008) has compared

human-human hand-over interaction with the same task done by a robot and a human, where reference (Lopez-Damian, Sidobre, Tour, & Alami, 2006) has presented a planning strategy to find grasps on objects for interactive manipulation tasks, e.g. handing-over task. By handing over the grasped object to the human, the proposed algorithms will assure that the human will find at least one place where to grasp the object. In general, the previously mentioned works have some drawbacks, e.g. the physical interaction during handing-over objects from human to robot is not perfectly handled. Furthermore, the transfer tasks are performed exclusively by the human, i.e. the robot will bring the robot hand into a specified pose and then it will wait until the human places the target object between the fingers of the gripper. When the robot detects that an object has been placed in its hand, it attempts to grasp the object. In fact, these strategies will not be fit to assist blind, disabled or elderly people or even to support workers concentrating on their work.

Another researches have focused on performing the handing-over task with the help of force/torque sensor, e.g. in (Kim & Inooka, 1992) and in (Mason & MaxKenzie, 2005) the behavior human's grasping is analyzed by force/torque sensor, i.e. the grip forces which are applied during the transferring of an object from one person to another are analyzed. This kind of investigation is indented to accomplish the smooth hand-over of an object between a human and a robot, where in (Nagata, Oosaki, Kakiura, & Tsukune, 1998) an experimental result is performed to describe the hand delivery of objects between a multi-fingered hand and a human based on force-torque sensing. Different oriented researches have focused on improving the recognition of hand posture in 3D intending to human robot interaction, e.g. (Kim, Kwak, & Chi, 2006), (Yin & Zhu, 2006) and (Chuang, Chen, Zhao, & Chen, 2011). The related contributions and improvements of this thesis will be deeply illustrated in chapter 5.3.

1.5 System setup

This work is supported with a wide range of different experimental applications. These experiments have proved the applicability of the proposed theoretical algorithms and they have insured the importance of combining vision and force information to perform complicate tasks e.g. improving impact force, library automation and human robot interaction.

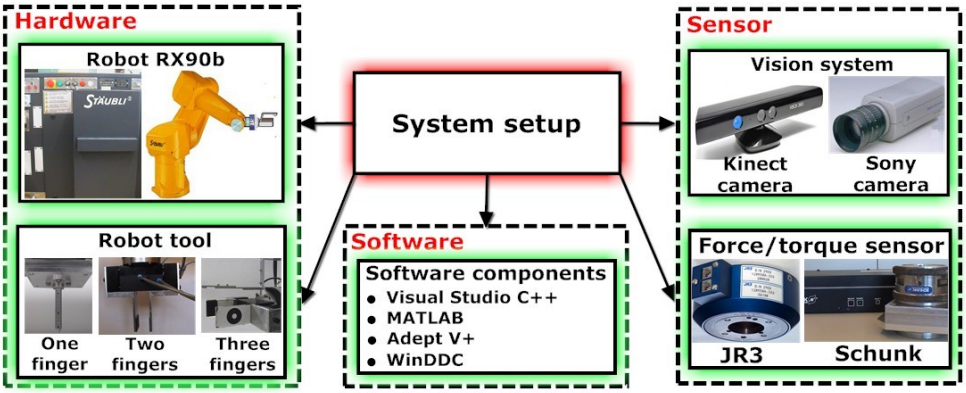

Fig. 1.30 Overview of system setup

The experimental systems consist of the following components:

1. Robot Stäubli RX90b
2. Robot tool
3. Vision system
4. Force/torque sensor
5. Software components

1.5.1 Robot Stäubli RX90b

Fig. 1.31 presents the Stäubli RX90b robot which is an industrial robot. The mechanical manipulator arm of this robot consists of stiff and lightweight robot links that are interconnected by means of six revolute joints. Every joint will connect two links together. The first and the second joints form the shoulder of the robot. The third Joint is the elbow and the fourth one is the forearm. The fifth and sixth joints make up the robot wrist. The motion of the robot is generated by six servo-drivers which drive six servo-motors. The servo motors are brushless three-phase servo motors. The feedback of the robot-position is attained by resolver. The nominal payload of the robot is equal to 6Kg.

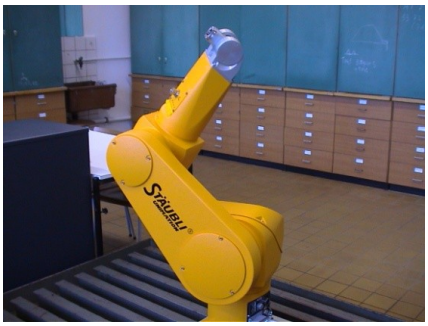

Fig. 1.31 Robot Stäubli RX90b

The robot is controlled by an industrial Staubli CS7b motion controller, see Fig. 1.32. This controller contains six independent motion controllers, which compute the current commands for digital current amplifiers located inside the CS7b controller. The term independent refers to the fact that every servo motor is equipped with a Single Input Single Output (SISO) controller. The inputs of the motion controller are the joint reference position and velocity.

Fig. 1.32 Robot controller

The robot controller encompasses the forward dynamic model of the robot arm and It computes the motion of the robot arm as a function of the joint torques. The motion of the robot arm is expressed in terms of a set of generalized degrees of freedom q (in RX90b they are six joint angles) and also their time derivatives $\dot{q}$ and $\ddot{q}$. The controller includes also the kinematic model of the robot which computes the actual Cartesian position and orientation, and time derivatives, of the robot as a function of the joint positions q and velocities $\dot{q}$.

In general, the modeling and identification of the industrial robotics, one of them RX90b, has been widely discussed. In (Wiaboer & Aarts, 2005) and (Hardeman, 2008), the modeling and identification of the robot RX90b have been deeply presented for laser welding (Waiboer, 2007). Another work (Khalil, Gautier, & Lemoine, 2007) has presented methods for the identification of the inertial parameters of the load of the RX90b manipulator with the help of SYMORO+ software (Khalil & Creusot, 1989). In the first experiment (improving the impact control), a new robot controller called WinDDC-real-time controller is implemented. This controller can control only 3 joints (q_2, q_3, q_5) of the robot system because the Jacobian matrix has been calculated only for them.

1.5.2 Robot tool

In this thesis three kinds of robot tool will be used which fit the proposed tasks; One-finger tool, two-finger gripper and 3-finger gripper.

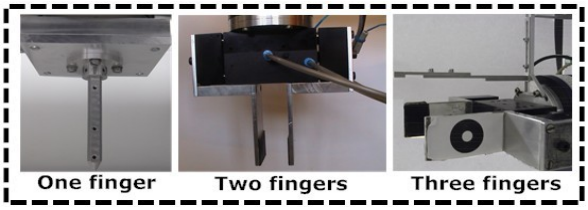

Fig. 1.33 Types of the implemented robot tool

One-finger tool is used to illustrate clearly the improving of impact control by combining vision and force feedback, where three-finger tool is used to presents the importance of combining force and vision feedback in order to grasp an object which is stuck between two others. Two-finger tool is normal gripper to grasp objects carried by human hand or located on flat surface.

1.5.3 Vision system

In this thesis two kinds of camera will be used, RGB camera for 2D visual servoing and RGBD camera for 3D visual servoing.

1.5.3.1 RGB camera (Sony DFW-X700)

Sony DFW-X700 is a fire-wire CCD color camera with 1024 x 768 pixel resolution. Its frame rate is 15 frames/s and the transfer rate is 400 Mbps with IEEE 1394 digital interface. This camera is easy to install and ideal for a variety of applications such as machine vision, image processing, microscopy and factory automation.

Fig. 1.34 Sony camera DFW-X700

CCD (charge coupled device) sensor moves the generated charge from pixel to pixel and converts it to voltage at an output node as is shown in (Fig. 1.35), on the contrary of CMOS principle (complementary metal oxide semiconductor) where the images convert charge to voltage inside each pixel. In general CCD technology has high quality images, low susceptibility to noise, good uniformity of the pixels, high fill factor.

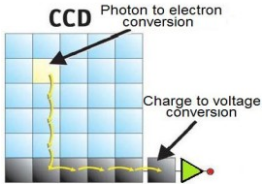

Fig. 1.35 CCD principle

The RGB camera contains a lens that forms a 2D projection of the scene on the image plane where the sensor is located. This projection causes direct depth information to be lost. Therefore, to control the robot in 3D using information provided by a RGB camera is needed to determine the 3D coordinates corresponding to an image plane point. This information may come from multiple cameras, multiple views of single camera or knowledge of the geometric relationship between several feature point on the target. Anyway, this camera is well suited to 2D robot applications.

1.5.3.2 RGBD camera (Kinect)

RGBD cameras are novel sensing systems that capture RGB images which every pixel of them is provided with depth information. Kinect camera is kind of RGBD camera which was originally intended to be a motion sensing input device for the Xbox 360. The Kinect sensor is a horizontal bar connected to a small base with a motorized tilt mechanism, designed to be positioned lengthwise above or below the video display. The hardware components of Kinect sensor are RGB camera, depth sensor, multi-array of microphones and tilt motor as shown in Fig. 1.36. The Kinect's depth sensor consists of an infrared (IR) light source and a laser that projects a pattern of dots that are read back by monochrome CMOS IR sensor. The sensor detects reflected segments of the dot pattern and converts their intensities into distances.

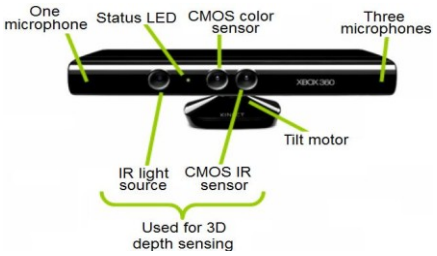

Fig. 1.36 Hardware of Kinect sensor

The most important features of Kinect sensor are:

- Each frame generated by RGB camera has VGA resolution (640 x 480 pixels).
- Each frame generated by the depth sensor has also VGA resolution (640 x 480 pixels) and it is containing 11-bit depth values which provide 2048 levels of sensitivity.
- The entire system operates at 30 frames per second
- The resolution of the depth dimension (along the z-axis) is about one centimeter
- The spatial resolution (along x- and y- axes) is in millimeters.
- The infrared laser projector uses a near-infrared light ”830nm” which is preferred because of suitable availability, low-cost sources and detectors and invisible to human-eye.
- Kinect is also reportedly able to simultaneously track up to six people at a time, including two active persons for motion analysis as well as feature extraction of up to 20 joints per everyone.

As shown in Fig. 1.37, Kinect sensor outputs two types of images 1. Depth image: which represents topographic view of the scene (matrix of pixels and each pixel contains a value representing the distance of the object from the sensor). 2. Color image: a standard output of a RGB 2D digital camera. To produce more accurate sensory information Kinect performs a process called registration. The registration process is resulting images which are pixel-aligned. In other words, every pixel in the color image is aligned to a pixel in the depth image.

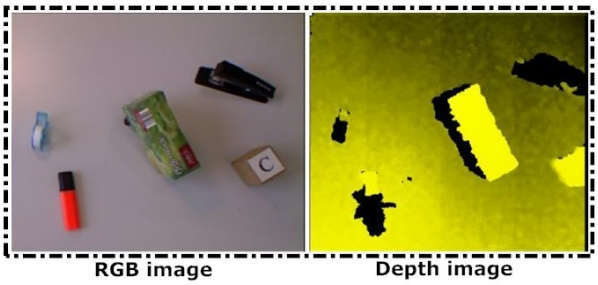

Fig. 1.37 Produced images by Kinect sensor

Previously, obtaining the texture and shape simultaneously in the real time using one-shot capturing was difficult. However, using a visible light camera and infrared light camera simultaneously solves this problem a paves the possibility to get more information about the object. More information about the concept of the Kinect sensor could be found in (Shupnt, Tikva, Zalesvsky, & Ha'ayin, 2008) and (Freedman, Shpunt, Machline, & Arieli, 2010). However, the next section will present briefly some explanations about main principles of 3D image devices for better understanding of Kinect camera.

1.5.3.3 Depth image Technology

Techniques to acquire the shape of an object are categorized into two types, passive methods like stereo vision, and active methods like laser range finders and structured light. A laser

range finders measure the distance between the sensor and an object based on triangulation or time-of-flight. Where, structured-light methods cast light pattern on an object using a projector and the resulting patterns are observed by a camera (Fofi, Sliwa, & Voisin, 2004) and (Albitar, Graebling, & Doignon, 2007).

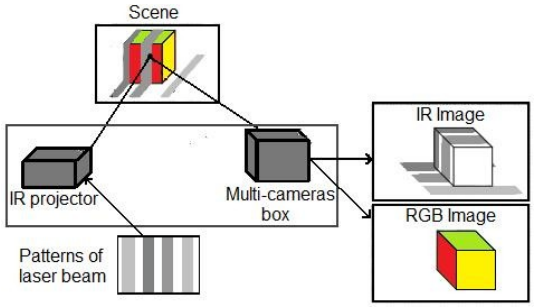

Fig. 1.38 Capturing IR and RGB images

As shown in Fig. 1.38 the patterns of laser beam are projected on the object, after that the IR & RGB images will be received with the help of Multi-cameras box. The distance from the camera to the object is computed with the active triangulation based on the correspondences to all pixels of the image. The major difficulty with metrical active triangulation is keeping the correspondence problem as simple as possible. This issue can be tackled by designing patterns which projected by light coding and then encoding them. The simplest patterns of laser beam could be single dot, single line, matrix of dots, stripes, circles etc. More complicated patterns are generated by projecting different sorts of patterns onto the scene to get the 3D map, e.g. one projector establishes structured light for comprising two sets of parallel stripes having different periodicities and angles. In another method, the object is illuminated with a matrix of discrete two dimensional pattern image, such as a grid of dots or grating pattern.

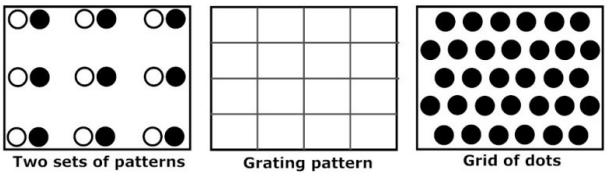

Fig. 1.39 Different types of patterns

Fig. 1.39 shows different types of patterns. In general, there are three schemes of pattern coding (Mutto, 2012):

- Direct coding: each pixel is represented by the pattern value at the pixel itself.
- Time-multiplexing coding: a sequence of N different patterns is projected to the surface. Then, each pixel will be represented by a sequence of the N pattern values.
- Spatial-multiplexing coding: The code is the spatial pattern distribution in a window of number of pixels which centered around the target pixel.

Each coding strategies has different advantages and disadvantages as described (Salvi, Pags, & Battle, 2004).The projected pattern of Kinect camera is considered as type spatial-multiplexing

approach and it is characterized by a horizontal uncorrelation of pattern. The term "uncorrelated pattern" refers to a projected pattern of spots, whose position are uncorrelated in planes transverse to the projection beam axis. The positions are uncorrelated in the sense that the auto-correlation of the pattern as a function of transverse shift is significant for any shift larger than the spot size and no greater than the maximum shift that may occur over the range of depths mapped by the system, such as pseudo-random (Morano, Ozturk, Conn, & S.Dubin, 1998) and quasi-periodic (Duneau & Katz, 1985). The advantage of these patterns is the low duty cycle.

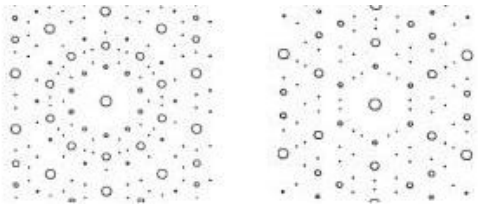

Fig. 1.40 Examples of uncorrelated patterns

The Image processor in the Kinect camera computes the 3D coordinates of point on the surface of object by triangulation. It is based on transverse shifts of the spots in an image of the pattern that is projected onto the object relative to a reference pattern at a known distance from device. In other words, the image processor compares the spatial-multiplexing windows which centered at each pixel of the captured image to the spatial-multiplexing windows in the reference image in order to find the most closely-matching windows of both images, after that the covariance between the spatial-multiplexing window centered at a reference pattern point and the one centered at the target point will denote the depth value. This means Δz will engender a concomitant transverse shift Δx in the spot pattern observed in the image.

To conclude, the depth value of points on the object may thus be determined by measuring shifts in the x-coordinates of spatial-multiplexing windows in the captured image relative to spatial-multiplexing windows in the reference image which is taken at a known distance z.

1.5.4 Force/torque sensor

The force/torque sensor is mounted on robot in order to measure the external forces and torques which are applied on it. In general, most manipulators are using 6 components force/torque sensor which is based on strain gauges.

Fig. 1.41 Force/Torque JR3 sensor

In our experiment two kinds of force/torque sensors are used. As shown in Fig. 1.41, the first one is six components JR3 (120M50A) force/torque sensor with internal electronic systems providing analog output (+5V to -5V representing +full load to –full load) for each axis (JR3-Web). Its effective measurement range is ± 100 N for force and ± 10 N.m for torque.

Fig. 1.42 Force/Torque Schunk sensor

The second one is produced by SCHUNK and it is called FT Delta SI-660-60 (SCHUNK-Web). The implemented SCHUNK force/torque sensor is supported with stand alone controller FTS. The FTS is connected to the sensor via RS-232 interface it converts the multiplex-form strain guage into forces and moments. Fig. 1.42 presents the SCHUNK FT-Delta force/torque sensor and its stand alone controller FTS (the black box). The effective measurement range of the force/torque sensor used is ±660 N for forces and 60 Nm for torques. Cartesian forces and torques are represented by analogue voltages (±5V for maximum/minimum force or torque).

1.5.5 Software components

The implemented softwares in this work are: Adept V+, WinDDC, MATLAB and Visual Studio C++. Adept V+ (Stäubli, 1992) is the programming language for the standard controller of Robot Stäubli RX90b. In addition to being the complete programming language, V+ is also a complete operating system that controls equipment connected to Adept controllers. The functional groups of V+ programming are: 1. Program instructions; 2. Functions; 3. System Parameters; 4. System switches (more details in (Stäubli, 1992)).

WinDDC software (Neumann, 1991) is a programming language which is designed for the new robot controller WinDDC-Real-Time- Controller, (Winkler & Suchý, 2005). WinDDC programming language contains commands enabling access to the peripherals and to software elements of control technology like integrators, differentiators, etc. The program development is performed with a standard PC (Windows operating system) and it can communicate with robot controller via serial connection. It is possible to keep influence on program variables during execution by PC and supervise these variables from the robot controller by their visualization. Furthermore, PC simulations are also possible using WinDDC.

In this work, MATLAB program and Visual Studio C++ are fully integrated with both robot programming languages, the standard V+ and the modified WinDDC. Adept V+. In other words, all robot commands could be written directly in MATLAB or Visual Studio programs.

Many libraries and toolboxes are implemented in the proposed program, e.g. OpenNI (OpenNi-Web), Prime Sense (Primesense-web), OpenCV, SIFT features (Lowe, 2004), CvBlobslib (Bloblibrary-web), Canny filter (Canny, 1986), Haar cascades filter (Pan, Ge, He, & Chen, 2009), Median filter (Hwang & Haddad, 1995), Progressive probabilistic Hough transform (Galambos, Matas, & Kittler, 1999), RGB filter, HSV filter, Morphological operations, Ethernet TCP/IP, etc. The functionally of these libraries will be illustrated in the next chapter.

Chapter 2

Image Processing

The vast majority of today's growing robot population operates in factories, offices and homes. The specialists have focused their efforts on the development of measurement instrumentations and sensors, because they have found that the first step to spread robot systems in all applications is when the measurement systems have higher performance, response and accurately. One of the common used sensors in robotic fields these days is vision sensor. Vision provides global information on surrounding environment to be used for motion planning and obstacle avoidance.

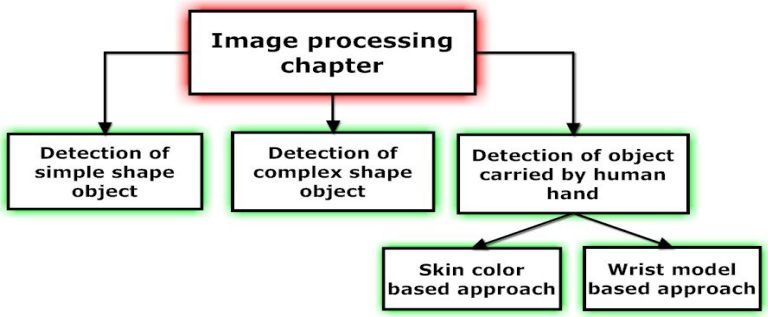

Fig. 2.1 Overview of Chapter 2

As previously mentioned, the main contribution of this work is improving the integration algorithms of vision/force control. This has encouraged us to develop some image processing algorithms intended to promote the vision/force integration. This work will handle with different types of objects, some of them could have simple geometric shapes and others could have complex form. Using the same image processing algorithm to detect different objects is impractical, because the algorithms should be implemented in the real time and the visual servoing is later required. Furthermore, if the object has complex structures, the same simple algorithm will not detect it. In addition to that, the same algorithm may not be useful for vision robot system if the object was carried by human hand. Hence, this chapter will propose three image processing algorithms to detect different kind of objects, as shown in Fig. 2.1. The first algorithm will be fit to detect the simple shape objects such as books, cubes etc. The second algorithm will detect different kinds of complex objects such as puncher, stapler etc. The last section will explain algorithms for detecting objects carried by human hand.

2.1 Detection of simple shape object

This section will propose an image processing algorithm to detect different kinds of simple shape coded objects. Those could be found in different places e.g. books in library, coded items in warehouses, stores, shelves or any simple shape object which has alphabetical/ numerical codification system. This section is supported with practical experiments implemented in an automated library scenario for detecting and sorting coded books.

2.1.1 Related work

Many papers have already worked on the problems of books detection and their label recognition, e.g. (Ramos-Garijo, Prats, Sanz, & Pobil, 2003) has proposed an assistant robot for book manipulation in library. This work computes the location of the books depending on their labels, after that it detects the vertical borders between the labels found in the image. The detected labels of the books will be processed using open source optical character recognition (OCR). The main limitation of this work is the arrangements of the books in the shelf should be always in a vertical position, i.e. the proposed vision algorithm is not able to detect the boundary of the books if they have inclination. Moreover, the vision system will not be able to detect the boundary of the label if it has the same color of the book. Furthermore, OCR software has some disadvantages, e.g. errors could sometimes happen such as misreading letters or skipping over letters, etc. In addition to that, OCR is very sensitive to affine transformations like scaling, translation, rotation, shearing or axis deformation and background color.

Another work (Tomizawa, Ohya, & Yuta, 2003) has proposed an algorithm to solve the problem of detection the sloping books by using a submask (rectangular area) which rotates from -45 to +45 degrees overlapping the area of the interest. The maximum number of matched pixels between the submask and the original image will refer to the current book location, so in this case the angle of the submask will be the degree of inclination of the book. However, the processing time of inclination measurement using this method is relative large. Furthermore, this work has not addressed the problems of labels recognition because it has assumed that the human can choose the target book remotely. Recently, another work (Heyer, Fragkopoulos, Heyer, & Gräser, 2012) has proposed algorithms for book detection which are based on vertical Hough line transformation. This work has assumed that all books should be in vertical position. Furthermore, the robot can grasp only the most right book from upper shelf because this work is not able to detect the code of the book. Hence, these drawbacks have made this work very limited.

In our work, the proposed vision algorithm detects objects' position/orientation, characterizes and classifies the objects, identify the codes assigned to objects (SIFT features which are invariant to image scale and rotation, affine distortion, changing in viewpoint, addition of noise and change in illumination). Furthermore, the vision algorithms will not only be used as a simple feedback or as desired position estimator but it will also extract the relations between the neighbor objects if they are stuck together or there is adequate space between them in

order to define the most appropriate vision/force control mode. The proposed algorithms will be able to detect the books and recognize their labels if they are vertical, stuck together or even if they have inclination degree.

2.1.2 Introduction

As shown in the Fig. 2.2, the steps of the proposed algorithm are as follows: After receiving the image from the vision system, it will be filtered by Canny filter and then it will be processed using some morphological operations. After that, the system will detect all edges in the image, calculate the junctions (intersection-points)/endpoints and segment the line to detect all connected lines which form the objects in the image.

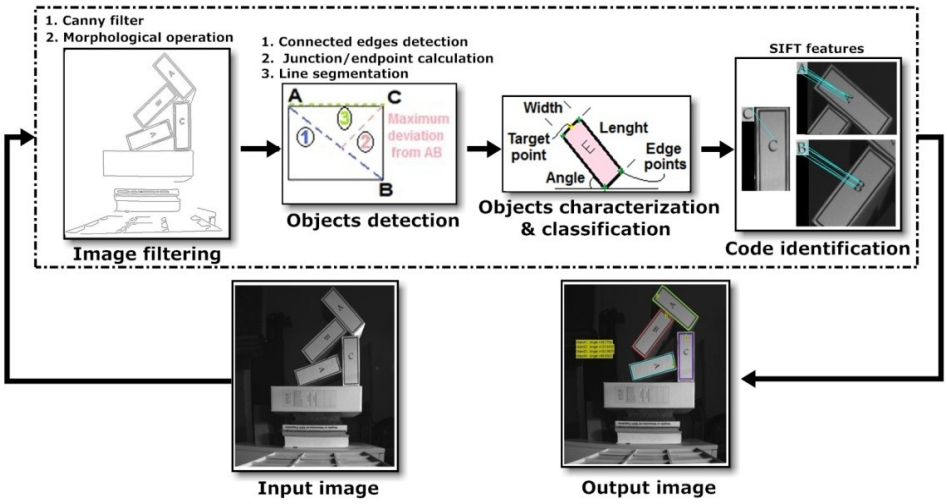

Fig. 2.2 Algorithms of simple shape object detection

Next steps characterize and classify the objects such as defining the length, width, corners, angles, shape etc. for every object and comparing them with target objects characteristics. After detecting the target objects the algorithm will identify every code of every target object using SIFT feature. Next section will illustrate these issues with more details.

2.1.3 Image filtering

In this approach, the target image will be converted from full color image $I(u,v)$ to gray image $G(u,v)$ and then to binary image $Bw(u,v)$ with the help of canny filter and some morphological operations.

2.1.3.1 Canny filter

Canny function (Canny, 1986) will take the grayscale image $G(u,v)$ as its input, and return a binary image $Bw(u,v)$ of the same size as $G(u,v)$ and $I(u,v)$, with 1's where the function finds edges in the grayscale image and 0's elsewhere (Fig. 2.3).

$$Bw(u,v) = Canny\_Filter(I(u,v)) \tag{2.1}$$

where (u, v) is the coordinate system in the image space. The binary image $Bw(u, v)$ will contain all the detected edges in the grayscale image $I(u, v)$.

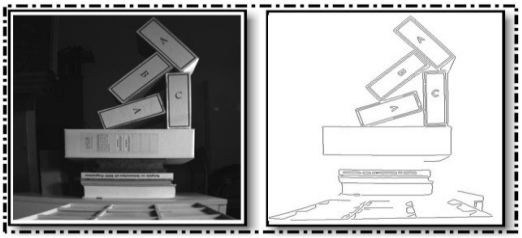

Fig. 2.3 Images before/after Canny filter

There are many methods to detect the edges in the grayscale image and convert them to binary image, such as Sobel method (Duda, 1973), Prewitt method (Prewitt, 1970), Laplacian of Gaussian etc. The main advantages of Canny filter that it looks for local maxima of the gradient. This gradient is calculated using the derivative of a Gaussian filter. Furthermore, Canny filter uses two thresholds, to detect strong and weak edges, and includes the weak edges in the output only if they are connected to the strong edges. Therefore it is less likely than the others to be fooled by noise and more likely to detect true weak edges.

2.1.3.2 Morphological operations

Some morphological operations will be performed, such as:

$$Bw(u, v) = Bwmorph(Bw(u, v), 'clean') \quad (2.2)$$

This instruction will remove the isolated pixels (individual 1's that are surrounded by 0's)

$$Bw(u, v) = Bwmorph(Bw(u, v), 'skel', Inf) \quad (2.3)$$

This instruction will remove the pixels on the boundaries of objects without allowing the objects to be broken (to make sure that the edges are thinned).

$$Bw(u, v) = Bwareaopen(Bw(u, v), P) \quad (2.4)$$

This instruction will remove all objects that have fewer than P pixels (removing the small objects). These instructions are performed in MATLAB and they will prepare the binary image to the next steps.

2.1.4 Objects detection

After using Canny filter and morphological operations the program will get the edges of the image. However, these edges are incomprehensible and meaningless. Therefore, in the next phase the program will link all connected points of edges together into lists, calculate the junction/endpoint and segments the edges' lines.

2.1.4.1 Connected edges detection

In this step the program will link all connected points of edges together into lists such as:

$$edge\_list = \begin{matrix} edge_1 \\ \begin{Bmatrix} [x_{11}\ y_{11} \\ x_{12}\ y_{12} \\ \dots\ \dots \\ x_{1Q}\ y_{1Q}] \end{Bmatrix} \end{matrix} \quad \begin{matrix} edge_2 \\ \begin{Bmatrix} [x_{21}\ y_{21} \\ x_{22}\ y_{22} \\ \dots\ \dots \\ x_{2R}\ y_{2R}] \end{Bmatrix} \end{matrix} \quad \begin{matrix} edge\ .. \\ \begin{Bmatrix} \\ \dots\ \dots \\ \\ \dots\ \dots \end{Bmatrix} \end{matrix} \quad \begin{matrix} edge_N \\ \begin{Bmatrix} [x_{N1}\ y_{N1} \\ x_{N2}\ y_{N2} \\ \dots\ \dots \\ x_{NT}\ y_{NT}] \end{Bmatrix} \end{matrix} \tag{2.5}$$

where $edge\_list$ is a cell array of edges in coordinate form where N is the number of discovered edges and Q, R and T are the numbers of edge points in every edge.

The program will start searching in $Bw(u, v)$ image for the first pixel which has value 1 from upper left corner; this pixel will be called start-pixel and its coordinate will be stored in the $edge\_list$ as (x_{11}, y_{11}). After that the program checks the value of the neighbor pixels to test if any of them is junction or endpoint and to define the next connected pixel which has the value 1. After ensuring that there is no junction or endpoint in neighbor pixels, the coordinate of the next connected neighbor pixel will be added to the list (x_{12}, y_{12}) and it will be the next center pixel. Then the program will repeat the previous steps and test the new neighbors again. If any of the neighbor pixels is junction or endpoint, this pixel will be added to the edge (x_{1Q}, y_{1Q}). The values of the pixels which are added to $edge\_list$ in the first edge will be converted to value 0 in order to not be used again. In the next operation a new edge (x_{21}, y_{21}) will be created and the previous steps will be repeated.

2.1.4.2 Junctions/endpoints calculation

In this program all pixels in the filtered image will be tested to find the junctions and endpoints of every edge. Fig. 2.4 presents examples for the start point (x_{11}, y_{11}), neighbor pixel, endpoint pixel and junction pixel.

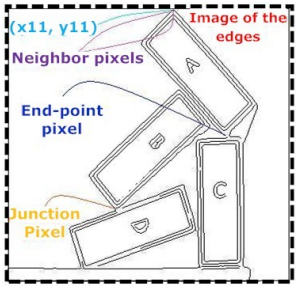

Fig. 2.4 image of the connected edges

As is known every pixel has 8 neighbors. The values of neighbor pixels will be tested in order to define the type of the center pixel e.g. endpoint, junction etc. Table 2.1 shows the indexing of the neighbor pixels:

1	4	7
2	Center pixel	8
3	6	9

Table 2.1 Address of the neighbors pixels

We assume a vector A, whereas $A = [x(1)\ x(2)\ x(3)\ x(6)\ x(9)\ x(8)\ x(7)\ x(4)]$, after left rotating of the pixels in vector A, we get $B = [x(2)\ x(3)\ x(6)\ x(9)\ x(8)\ x(7)\ x(4)\ x(1)]$, According to the following equation (2.6) one can see, which pixel represents junction or endpoint.

$$V = \sum_{i=1}^{8} |A(i) - B(i)| \qquad (2.6)$$

If $(V \geq 6)$ this means that three neighbor pixels have the value 1 around the center pixel, i.e. the center pixel is junction as shown in the Table 2.2.

0	0	0
1	Center pixel	1
0	1	0

Table 2.2 Example of junction point

If $(6 > V > 2)$ this means the center pixel is connected pixel as shown in Table 2.3

0	0	0
1	Center pixel	1
0	0	0

Table 2.3 Example of connected pixel

If $(V = 2)$ this means that only one neighbor pixel has the value 1 around the center pixel, i.e. the center pixel is endpoint as shown in the Table 2.4.

0	0	0
0	Center pixel	1
0	0	0

Table 2.4 Example of endpoint pixel

2.1.4.3 Line segmentation

In this phase, the program will take all the points in every edge, find the size and position of the maximum deviation from the line that joins the start-point, if the maximum deviation exceeds the allowable tolerance of line deviation, the edge is shortened to the point of maximum deviation and the test is repeated. In other words, each edge will be broken down to line segments. Assume that in Fig. 2.5 is the first discovered edge which will be segmented to lines. Initially, the program will calculate the point which has the maximum Euclidean distance from the start point A according to equation:

$$Max_{Euc} = (\sqrt{(x - x_{11})^2 + (y - y_{11})^2}) \tag{2.7}$$

It is clear that B has the maximum Euclidean distance, which will be considered as the temporary end-point of the line. Here (x, y) is the vector of coordinate which contain the all points of the first edge, (x_{11}, y_{11}) are the coordinate of the first point of the first edge.

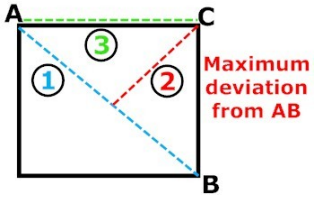

Fig. 2.5 Steps of line segmentation

Next step will calculate the maximum deviation of the edge-points from the line AB. As is known the equation of the straight line can be written as follows:

$$\frac{x - x_B}{y - y_B} = \frac{x_A - x_B}{y_A - y_B} \tag{2.8}$$

By rearranging the previous equation:

$$x \cdot (y_A - y_B) + y \cdot (x_B - x_A) + y_B \cdot x_A - y_A \cdot x_B = 0 \tag{2.9}$$

All the points belong to this line could be compensated in equation (2.21). Hence, the deviation of all edge-points which do not belong to this line AB will be calculated using equation:

$$dev = x \cdot (y_A - y_B) + y \cdot (x_B - x_A) + y_B \cdot x_A - y_A \cdot x_B \tag{2.10}$$

It is found by calculation that C has the maximum deviation from the line AB. If this maximum deviation is greater than the given threshold of the line deviation (a constant which is the maximum deviation from straight line before breaking a segment into two lines), the line AB will be shortened to the point of the maximal deviation. By adjusting the end-point from B to C, the new line segment is AC and the whole previous operation will be repeated until maximum deviation is smaller than the constant threshold of the line. Hence, at the end of this phase all objects in the image will be defined and segmented to lines as follows:

$$edge\_list = \begin{matrix} edge_1 & edge_2 & edge.. & edge_N \end{matrix} \begin{Bmatrix} [x_{11}\ y_{11} & [x_{21}\ y_{21} & \cdots\cdots & [x_{N1}\ y_{N1} \\ x_{12}\ y_{12} & x_{22}\ y_{22} & & x_{N2}\ y_{N2} \\ \cdots\cdots & \cdots\cdots & & \cdots\cdots \\ x_{1E}\ y_{1E}] & x_{2F}\ y_{2F}] & \cdots\cdots & x_{NK}\ y_{NK}] \end{Bmatrix} \tag{2.11}$$

Here N is the number of discovered edges, E, F and K are numbers of lines in every edge. (x_{11}, y_{11}) is the coordinate of the start-point of the first line in the first edge. (x_{12}, y_{12}) is the coordinate of end-point of the first line in the first edge which is also the start-point of the second line. (x_{1E}, y_{1E}) is the coordinate of the end-point of the last line in the first edge, which is also the start-point of the first line (x_{11}, y_{11}) if the edge is closed contour.

Important question

One could ask, why we didn't use line Hough transform in order to detect such simple lines?

Answer

Actually line Hough transform is an optimal algorithm to detect a line, if all the pixels are located exactly on the same straight line without any drift/shifting error caused by converting or filtering the image.

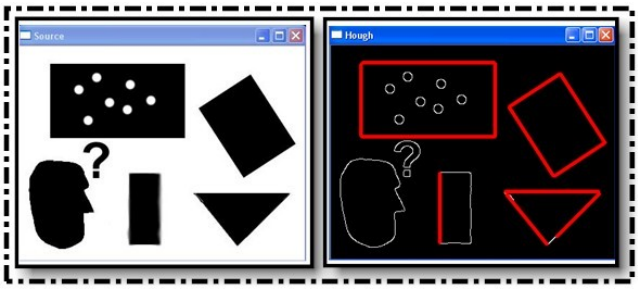

Fig. 2.6 Line Hough transformation

As shown in Fig. 2.6, line Hough transform can detect the lines in the big rectangle perfectly. However, as shown in case of the small rectangle line Hough transform will not detect its lines because of pixel shifting or it will segment every line to small lines.

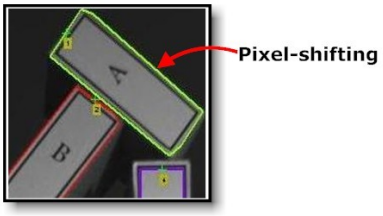

Fig. 2.7 Line detection using the proposed algorithm

Fig. 2.7 presents real results of lines detection with the help of the proposed algorithm. In the case of sloping lines, the line Hough transform will not be perfectly performed, whereas the proposed algorithm is able to detect this lines even if a pixel-drift has occurred during converting from color image to binary image and using canny filter.

2.1.5 Objects characterization and classification

In the previous phases, all connected components (objects) in the binary image $Bw(u, v)$ are detected and then they are segmented to lines. The start/end-points of all lines in every object are known and listed in equation (2.11). In this block all founded objects will be handled in order to find their shapes, sizes and some other characteristics, and then to compare this information with the characteristics of the target object looked for. Calculating of the length, angle and coordinate of the middle point of every line will be performed using equation:

$$length = \sqrt{(x_{start} - x_{end})^2 + (y_{start} - y_{end})^2} \qquad (2.12)$$

$$Tan\_Angle = \frac{(y_{start} - y_{end})^2}{(x_{start} - x_{end})^2} \tag{2.13}$$

$$(x, y)_{mid.point} = \left(\frac{(x_{end} + x_{start})}{2}, \frac{(y_{end} + y_{start})}{2}\right) \tag{2.14}$$

Using the results of the previous equations, the program will list this information in a matrix which includes all the characteristics of all objects in the binary image as follows:

$$Object_{Mat} = \begin{bmatrix} Obj_{wid}(1) & Obj_{leng}(1) & Obj_{ang}(1) & Obj_{PntY}(1) & Obj_{PntX}(1) \\ & & \dots & & \\ Obj_{wid}(n) & Obj_{leng}(n) & Obj_{ang}(n) & Obj_{PntY}(n) & Obj_{PntX}(n) \end{bmatrix} \tag{2.15}$$

Here $Object_{Mat}$ is characteristic matrix, n is the number of the objects, Obj_{wid} is the object width, Obj_{leng} is the object length, Obj_{ang} is the object angle and (Obj_{PntX}, Obj_{PntY}) are the coordinates of the target point of the object. These characteristics are illustrated in Fig. 2.8:

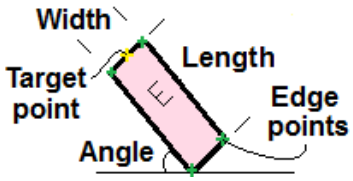

Fig. 2.8 Characteristics of the object

The detected objects can be classified according to the shape, number of lines, number of corners, number of right angles size of object etc. The program will look for all the objects which have the same characteristics of the books. As known, the general characteristics of the books, that they have four lines, every two of them are parallel and have the same length (rectangle shape). Hence, in this case the program will test all the characteristics of objects if they meet the conditions of being the book.

Searching algorithms should be thorough and accurate, because there are a lot of objects or edges in the room that could meet the conditions of a book but they don't have any meaning or are not important, see e.g. forms in Fig. 2.9.

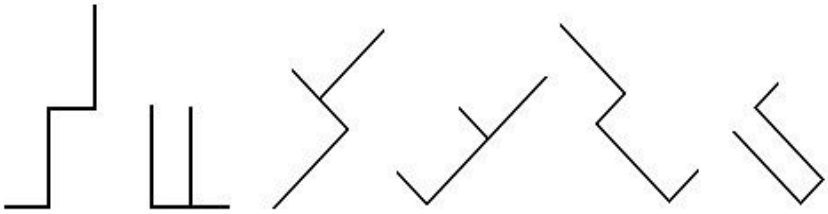

Fig. 2.9 Some shapes are similar to the rectangle

At the same time searching algorithm should have certain error permittivity, because sometimes the light is unevenly distributed on all sides of the objects, this may lead that the lengths of the lines or the angles are not completely equal after using Canny filter.

After detection all the objects which have the characteristics of the book, the program will list this object in matrix as follows:

$$Object_{List} = \begin{matrix} Object1 & Object2 & & Object\ m \\ \left\{\begin{bmatrix} x11 & y11 \\ \dots & \\ x1E & y1E \end{bmatrix}\right. & \begin{bmatrix} x21 & y21 \\ \dots & \\ x2F & y2F \end{bmatrix} \dots & \dots \begin{bmatrix} xM1 & yM1 \\ \dots & \\ xMK & yMK \end{bmatrix}\left.\right\} \end{matrix} \quad (2.16)$$

Here $Object_{List}$ is a list matrix which contains the corner points of all objects which meet the conditions of the target objects' characteristics; m is the number of target objects in $Bw(u, v)$ image.

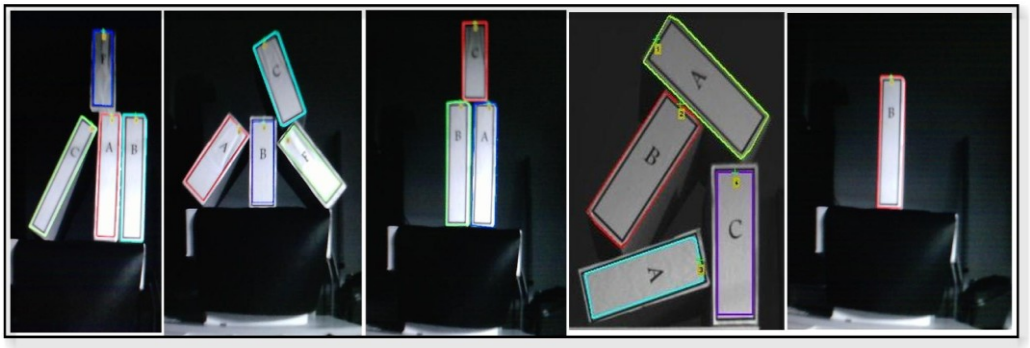

Fig. 2.10 Experimental Results

Fig. 2.10 presents the experimental results on different books with different poses and it shows the good performance of the proposed vision algorithms even if the books are in a vertical position, stuck together or even if they have inclination degree. The proposed algorithm was successfully implemented and tested in a grayscale image (without using any color features or histogram). Hence, in the future it can be combined with some color based image processing algorithms to enhance the performance of boundary detection.

2.1.6 Code identification

It is assumed in this work that every object contains codification system; this codification system consists of numbers and letters, which called alphabetical/numerical codification system. The codification system will be detected by the same camera which detects the target objects. Code identification will be performed using SIFT features.

SIFT algorithm (Lowe, 2004) is an algorithm in computer vision to detect and describe local features in images. SIFT algorithm extracts the interesting points of any objects in an image, which called "features description". SIFT method extracts distinctive invariant features from images that can be used to perform reliable matching between different views of an object or scene. These features are invariant to image scale and rotation, affine distortion, changing in viewpoint, addition of noise and change in illumination.

All these advantages make the SIFT features an optimal solution for recognition of letters and numbers on arbitrary placed and oriented objects rather than other methods such as OCR

(optical character recognition). Actually there are different features descriptors which are invariant to image scale and rotation, affine distortion, changing in viewpoint, addition of noise, and change in illumination such as PCA-SIFT, SURF (Bay, Ess, Tuytelaars, & Gool, 2008) etc. However, according to some researches such as (Juan & Gwun, 2009) and our experimental results it found the following: Although SIFT features descriptor is slightly slower than the others but it is more stable to the changes which occur in rotation and scale. Furthermore, usually SIFT method is optimal to use in detecting the object or scene which has complicated texture or huge amount of key-points, in this case the results are very accurate but they require sometimes a long cycle time of calculations. In the case of letters and numbers, the texture of them is simpler. Hence, the search process will take a very short time (few milliseconds for every character). Although there is great rush among researchers to use SIFT feature in recognizing 3D objects and in matching the scenes, however according to our knowledge no one has used it to recognize alphabetic/numeric codification system of items.

According to experimental results, the average number of detected key-points in each character ranges from 3 to 10 and they are different from one character to the other. In this work the letters and numbers are divided depending on the accuracy of classification into three groups as shown in Table 2.5.

Group A **Key-points (3-10)**					**Group B** **Key-points < 3**				**Group C** **common key-points**
A	B	E	F	G	C	D	I	L	C G
H	J	K	M	N	O	P	S	V	O Q
Q	R	T	W	X	0	1	2	7	E F
Y	Z	3	4	5					I 1 L J
6	8	9	ϑ	π					6 9
φ	&	§							P R

Table 2.5 Groups of letters according to possibility of classification

Group A contains all characters which will be very well classified because the structure of these characters consists of intersecting curves. This leads to situation that each character has sufficient number of key-points for classification.

Fig. 2.11 shows some example images of different letters. The first example shows two images of the letter A, the first one is the reference image and the second one is the real image of a group of different letters. The blue lines connect the common key-points of the letter A in both images. As shown in Fig. 2.11, the classification of the letters and numbers of group A works

very well without any errors, even if the letter has been rotated, scaled or even if the viewpoint of the camera and the illumination conditions have been changed.

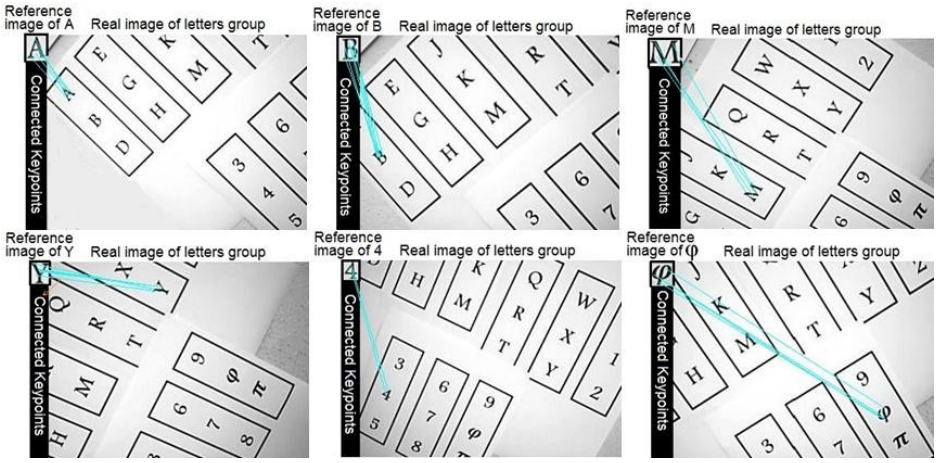

Fig. 2.11 Some examples of letter classification

Group B includes all the characters which have very simple structure (number of key-points are less than 3 features), which leads to difficulties in classification. Fig. 2.12 shows the only key-point of the letter C.

Fig. 2.12 One key-point of letter C

Group C illustrates different characters which have similar parts or common key-points as is shown in Fig. 2.13.

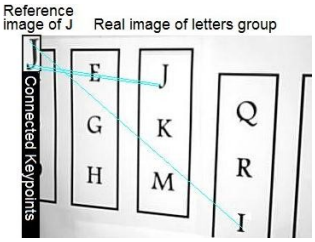

Fig. 2.13 Common key-point between J, I

In Fig. 2.13 there are two blue lines connecting common key-points of letter J in the reference image and in real image but there is also one wrong blue line between letter J and I because they have one common key-point. It can be summed up that the group A contains the characters, which can be used in the codification system, if high accuracy in character classification system is required. However, the codification system should avoid using similar characters (Group C), such as letter R and letter P or number 6 and number 9. Hence, the

characters which could be used with satisfied results are 16 letters, 5 numbers and 5 extra special letters. Usually the number of characters in the codification system for every material ranges from 7 to 9 characters. The total possibilities of forming different codes can be calculated as follows:

$$P_n^k = \frac{n!}{(n-k)!} \tag{2.17}$$

where P_n^k is the total possibilities of forming different codes, n is the total characters which can be used in the code and k is the number of characters in every code. Hence, to calculate the total possibilities of forming different codes using our alphabetical/numerical codification system and with the help of SIFT features as detector, it can be written as follows:

$$P_{26}^9 = \frac{26!}{(26-9)!} \approx 113 \cdot 10^{10} \tag{2.18}$$

As shown in Fig. 2.10, the implemented book label consists only of one letter and it is not in the real size of the book labels in the library. The classification of label consisting of more than one character or number will work also in the same efficiency, because the same principle will be used. However, the using of character which has larger size than the normal book label was only to simplify the hardware equipment. It can be easily solved by using camera which has more resolution and greater zoom factor.

2.1.7 Connecting labels with the related books

Another problem could be appeared, when the books location is unknown, e.g. they are stuck together in vertical position or they have inclination degrees. This problem could be summarized as follows: How the vision system will ensure which label (codification system) belongs to which book. In other words, how the system will define all the characters and numbers which are inside the boundary of the book. Especially that, our proposed system detects the book boundary separately from its labs and without any limitation or assumption about the location of the books or their labels.

As is known by any given three points in a plane, e.g. in Fig. 2.14 (a_1, a_2 and p_1), the area of the triangle determined by them is given by the following equation:

$$T_{area} = \frac{1}{2}\begin{vmatrix} a_{x1} & a_{y1} & 1 \\ a_{x2} & a_{y2} & 1 \\ p_{x1} & p_{y1} & 1 \end{vmatrix} \tag{2.19}$$

The value of the expression above is:

$$T_{area} = \frac{1}{2}(a_{x2} \cdot p_{y1} - a_{y2} \cdot p_{x1} - a_{x1} \cdot p_{y1} + a_{y1} \cdot p_{x1} + a_{x1} \cdot a_{y2} - a_{y1} \cdot a_{x2}) \tag{2.20}$$

T_{area} will be positive if the three points are taken in a anti-clockwise orientation, and negative otherwise. Defining whether if the label locates inside the book boundary or outside will be as follows: To be one pixel inside a rectangle (or any convex body), as we trace around in a anti-

clockwise direction from $a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow a_4$ and back to a_1, the areas (which shown in equation (2.20)) of triangles $a_1 a_2\, p_1$, $a_2 a_3\, p_1$, $a_3 a_4\, p_1$ and $a_4 a_1\, p_1$ must all be positive.

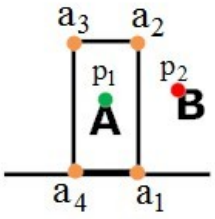

Fig. 2.14 Related label

When the direction of the points is not known if it is anti-clockwise or clockwise, then the area of triangles must be either all positive or all negative. Whereas, when the label is locates outside of the book boundary, e.g. character B, some areas will be negative and the others will be positive. In this way, we can define all characters which are belong to the target book.

The calculations of the graspability, how the robot will grasp the books and the relation between neighbor books will be also based on equations (2.19) and (2.20) and they will be illustrated in details in chapter 4.

2.1.8 Conclusion

In this section, the proposed vision algorithm has been illustrated for detecting different kinds of simple shape objects. This algorithm detects objects' position/orientation, characterizes and classifies them and then identifies the codes assigned to objects using SIFT features. The proposed vision algorithm has shown very good performance for detecting the books boundary and for recognizing their labels without any limitations; if they are vertical, stuck together, inclined, etc. and even if the illumination or viewpoint have been changed. In addition to that, using SIFT features to detect the label has illustrated a better efficiency than previous work which has used ORC algorithm. The cycle time of the proposed algorithm could range between 1 sec and 1.5 sec per frame depending on number of characters in the image and the image size. Furthermore, the vision algorithms will also extract the relations between the neighbor books if they are stuck together or if there is sufficient space to enter the parallel gripper between them (see Chapter 4). The proposed algorithm could be easily modified to be implemented for detecting any simple shape coded items in warehouses, stores, shelves, etc.

2.2 Detection of complex shape object

The previous algorithm (simple shape object detection) will not perfectly detect the complex shape objects such as puncher, stapler etc., because these objects consist of many intersected curves and lines, the previous algorithm was optimal to detect only the simple shape object such as books, cubes etc. in real time. This motivates us to propose another algorithm to detect objects which have complicated geometry and their forms have many intersected curves and lines. This section will illustrate the proposed detection algorithm of complex shape objects and it will present the integration of different image processing algorithms with the help of Kinect camera to detect objects with no need to any priori model of them.

2.2.1 Related work

In the recent years, Kinect camera has been increasingly used in different applications, especially in indoor and service robot applications e.g. (Filliat, Battesti, Bazeille, & Meyer, 2012) and (Stampfer, Lutz, & Schlegel, 2012). Reference (El-laithy, Huang, & Yeh, 2012) has presented an overview about the advantages and disadvantages of using Kinect camera for robotics applications. The proposed image algorithm in this section will be implemented to handle the problems of detecting unknown objects placed on flat area e.g. table or conveyor. Some previous works have used template matching algorithms to detect unknown objects e.g. (Tombari & Stefano, 2010) and (Xiao, Hu, Gao, & Wang, 2010). However, such approaches require scanning of the objects as a priori knowledge which is not always possible. Furthermore, as the number of the objects increases, so does the matching time for identifying the target item in the scene which is not suitable for real time applications.

Another possibility to deal with model-free objects is to approximate them by primitives like boxes or cylinders e.g. (Collet, Berenson, Srinivasa, & Ferguson, 2009), (Nieuwenhuisen, Stückler, Berner, Klein, & Behnke, 2012) and (Goron, Marton, Lazea, & Beetz, 2012). On the contrary of the approaches requiring particular geometric properties or CAD models, these approaches require a learning phase to create the 3D approximated model of the objects from their natural features. Another work (Baumgartl & Henrich, 2012) has proposed robot system which is able to grasp unknown objects with fast cycle time ca. 36ms. This work has used Hough transformation to extract the edges of the objects and then grouped these edges together. Actually in the real application, there is no guarantee to produce a closed contours or edges for every object, especially when the illumination is permanently changing and it is unequally distributed on the scene. Hence, one object can be segmented into many separated regions.

The proposed algorithm in this work will combine different algorithms in image processing together in order to segment and detect unknown objects in real time, even if the illumination is changed, even if these objects have complicated shapes or they have bad contours with no need to a priori model of them.

2.2.2 Introduction

In this section, the object detection will be performed using the Kinect camera and it will benefit from both images $I_{RBG}(u,v)$ and $I_{depth}(u,v)$. In the Kinect camera, there is small translation between RGB image and depth image, as shown in the Fig. 2.15. The same point of the real world has two different positions in images coordinate system.

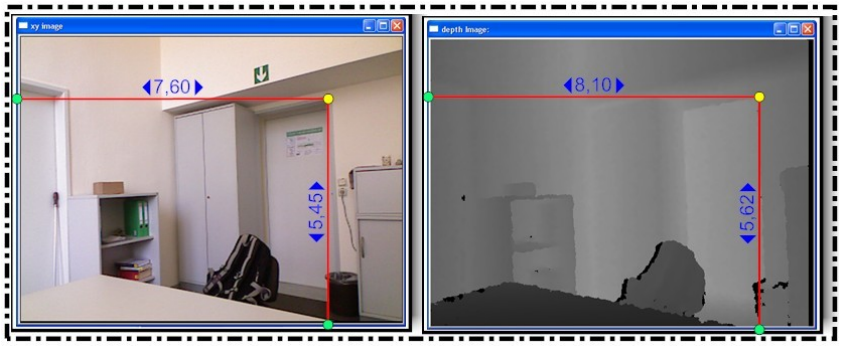

Fig. 2.15 RGB and Depth images before registration

This translation could be compensated by using the following function in OpenNI library:

$$depth.GetAlternativeViewPointCap(\ \ ).SetViewPoint(I_{RBG}(u,v)\ )$$

This function sets the view point of the depth image generator to match the RGB image generator. This process is called the registration, which means that every point in the real world will be on the same pixel in the RGB and depth images as shown in Fig. 2.16.

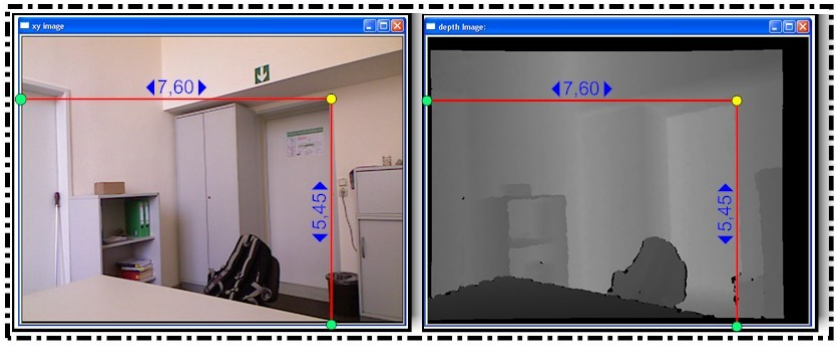

Fig. 2.16 RGB and depth images after registration

First of all, the RGB image will be used to detect the target object in 2D image. This phase consists of two steps: object segmentation (blob segmentation) and object classification (matching SIFT features). In other words, the system will segment all the objects in the RGB image then find the object (segment) which contains the most SIFT features. The resulting quality of segmentation is very important, because whenever the system gets a correct segmentation results it can estimate the object pose more accurately. Hence, this work has divided the object segmentation into many steps.

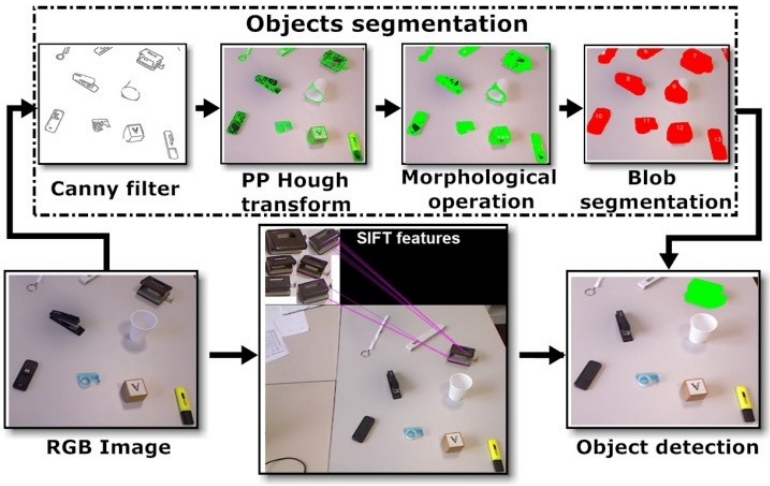

Fig. 2.17 Steps of object detection

Fig. 2.17 shows the main steps of object segmentation and detection which are: Canny filter, progressive probabilistic Hough transform and some morphological operations.

2.2.3 Objects segmentation

In general, image segmentation involves the calculation of local derivatives of the image. As known, the first step of image segmentation is the contour detection. Whenever the contours are perfectly detected, the objects of the image will be more precisely segmented.

Many methods have been proposed to find the edges using derivative approximation such as Sobel, Prewitt etc. As shown previously, Canny filter is the best method and it will take the grayscale image as input, and returns a binary image with same size as the original image (1s where the function finds edges in the grayscale image and 0s elsewhere).

Fig. 2.18 Converting from RGB to binary image

Canny image will serve as input image for the next phase which is called the preparation phase. Preparation phase will be necessary to improve the contour of the objects in order to perform the segmentation phase successfully. The first step of preparation phase is to detect all the lines in Canny image with the help of progressive probabilistic Hough transform (will be illustrated in the next section). After that the width of the detected lines will be enhanced by multiplication. In the next step dilating morphological operation will be performed in order to fill the small holes inside the objects.

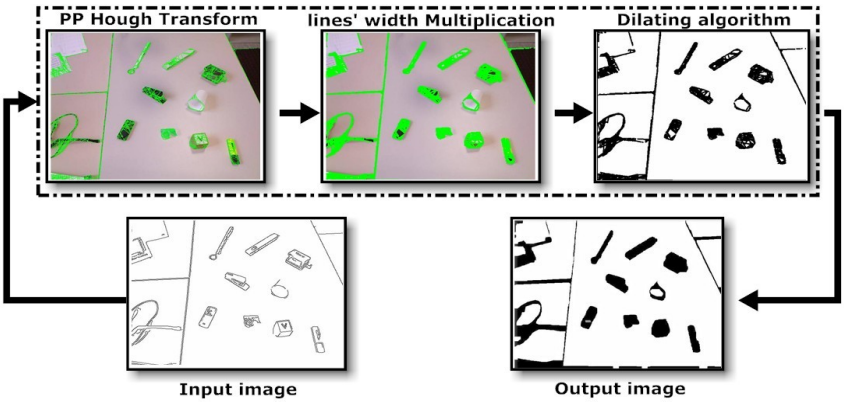

Fig. 2.19 Preparation phase for segmentation

2.2.3.1 Progressive probabilistic Hough transform (PPHT)

The Hough Transform (HT) is a popular method for the extraction of geometric primitives such as lines. The idea of the Hough transform is that every edge point in the edge map will be transformed to all possible lines that could pass through that point. Line equation can be written as follows:

$$r = x \cdot cos\theta + y \cdot sin\theta \tag{2.21}$$

$$y = -\frac{cos\theta}{sin\theta} \cdot x + \frac{r}{sin\theta} \tag{2.22}$$

The parameters θ and r is the angle of the line and the distance from the line to the origin respectively. By transforming from (x, y) coordinate to (r, θ) coordinate every line will be represented by a single point.

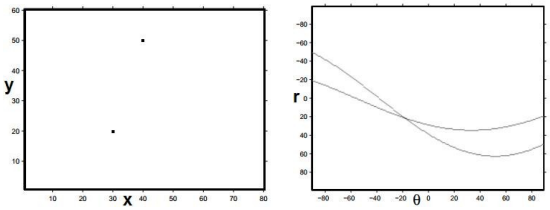

Fig. 2.20 Transforming to Hough space

Left image shows the points p_0 and $\mathrm{p}_1$in (x, y) coordinate, right image shows all possible lines through p_0 and p_1 represented in the Hough space

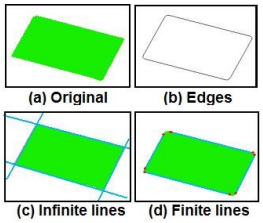

Fig. 2.21 Line Hough transform

The standard Hough transform detects lines given by the parameters r and θ and without any information about their lengths. All detected lines are of infinite length. If finite lines are required, some additional analysis must be performed to determine which areas of the image contribute to each line. Many algorithms have been suggested, one of them is to store coordinate information for all points in the accumulator, and use this information to limit the lines. However, this would cause the accumulator to use much more memory.

In general, Hough transform is not a fast algorithm for finding infinite lines in images of a certain size. Since additional analysis is required to detect finite lines, this is even slower. A way to speed up the Hough transform and find finite lines at the same time is the progressive probabilistic Hough transform (Galambos, Matas, & Kittler, 1999). The main idea of this method is to transform randomly selected pixels in the edge image into the accumulator. When a bin in the accumulator corresponding to a particular infinite line has got a certain number of votes, the edge image is searched along that line to see if one or more finite line(s) are present. Then all pixels on that line are removed from the edge image. In this way the algorithm returns finite lines. If the vote threshold is low the number of pixels to evaluate in the accumulator gets small.

Important question

The main question here is: Is the proposed preparation phase necessary to accomplish the segmentation task successfully?

Answer

In general, contour detectors offer no guarantee that they will produce closed contours, especially when the illumination is permanently changing and it is unequally distributed in the scene. Hence, they do not necessarily provide a partition of the image into regions. In other words, if there is any error in contour detection, it will affect performance of region segmentation. In our approach, the implemented PPHT is not used to detect the object contours but it is intended to detect every line in the object in order to enhance them and after that to help the system to fill the object region.

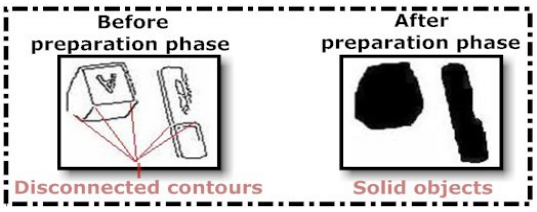

Fig. 2.22 Necessity of preparation phase

Fig. 2.22 presents two objects (cube and color marker) and it compares both images before and after the preparation phase. As shown, the first image consists of disconnected contours, which means that every object could be segmented into small regions, whereas the objects in the second image are presented as solid objects. Hence, every object in the image will be segmented as one region (segment).

2.2.4 Blob segmentation

After preparation phase, the system will able to segment the objects successfully. Segmentation phase will be performed with the help of CvBlob library. CvBlob is a computer vision library for labeling the connected regions and components in binary digital images and it also provides functions for manipulating, filtering and extracting the features from the extracted blobs. Hence, after performing Canny filter and the preparation phase on the current image $I(u, v)$ the CvBlob library will be used as follows:

First instruction: $CBlobResult\ blobs;$

This instruction defines blobs class which calculates the blobs of the current image $I(u, v)$ and allows later to extract the properties of them or select them according to certain condition.

Second instruction: $Blobs\ =\ CBlobResult(I(u, v), Null, 0);$

This instruction will extract all the blobs of the image $I(u, v)$

Third instruction:

$$Blobs.Filter(Blobs, B\_EXCLUDE, FilterAction, Condition, LowLimit, HightLimit);$$

This class provides functions to filter the blobs using some conditions such as area, length conditions etc.

$$Blobs.Filter(blobs, B\_EXCLUDE, CBlobGetArea(), B\_LESS, Parameter1);$$

$$Blobs.Filter(blobs, B\_EXCLUDE, CBlobGetArea(), B\_GREATER, Parameter2);$$

The last two instructions will filter the blobs in the image $I(u, v)$ with respect to the area conditions. In other words, all blobs of which areas are less than Parameter1 and greater than Parameter2 will be deleted from the CBlobResult (Blobs) class.

Fig. 2.23 shows all the detected blobs (segments) $S_i(u, v)$, in the current image $I(u, v)$. Here i is the number (label) of detected segments and $S_i(u, v)$ is a matrix which represents all the pixels belonging to the segment i, where $S_i(u, v) \in I(u, v)$.

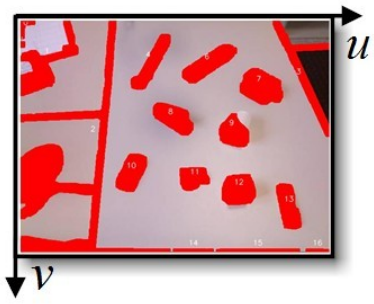

Fig. 2.23 Blob segmentation

By using CvBlob library we can also extract some features of the blob i such as moments, centroid, length etc.

$$lobs.GetNumber(i, CBlobGetMoment(0,0));$$

$$Blobs.GetNumber(i, CBlobGetXCenter);$$

$$Blobs.GetNumber(i, CBlobGetLength);$$

2.2.5 Objects classification

SIFT algorithm as shown previously extracts features which are invariant to image scale, rotation, affine distortion, changing of viewpoint, addition of noise and change in illumination. Matching the SIFT features will be performed between two different images: The reference image which contains different views of the desired object and the current image $I(u, v)$. To perform reliable recognition, it is important that relative positions between the features extracted from the training image in the original scene shouldn't change from one image to another. Similarly, features located in flexible objects would typically not work if any change in their internal geometry would happen between two images.

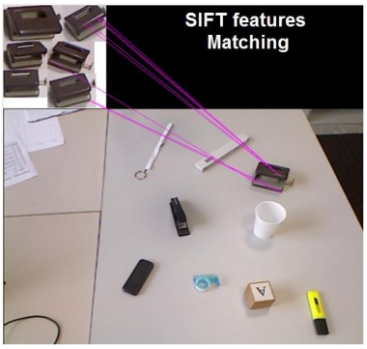

Fig. 2.24 Matching SIFT features

The output of SIFT algorithm will be vector $\vec{F}_{u,v}$ containing the values of u and v in the current image $I(u, v)$ for all the features (key-points) which are matched in the reference image and the current image as shown in Fig. 2.24. The Violet lines connect the common key-points of the desired object in both images.

At the same time, using some functions of CvBlob library, the calculation of minimum and maximum pixels for every blob will be performed as follow:

$$Max\_Vector[i].u = Blobs.GetBlob(i) -> MaxX();$$

$$Max\_Vector[i].v = Blobs.GetBlob(i) -> MaxY();$$

$$Min\_Vector[i].u = Blobs.GetBlob(i) -> MinX();$$

$$Min\_Vector[i].v = Blobs.GetBlob(i) -> MinY();$$

where $Max\_Vector[i]$ is an array containing the values of maximum u and v which belong to the blob i. $Min\_Vector[i]$ is an array which contains the values of minimum u and v belonging to the blob i.

As the last step, the system will define the blob with highest number of detected SIFT features in order to define the desired object. If a SIFT feature k with position (u_k, v_k) locates between (u_{max}, v_{max}) and (u_{min}, v_{min}) of blob i, it will belong to the blob i.

$$if\ \left(F_{u,v}(k)\ \in\ S_i(u,v)\right)\ then$$
$$T(i) = T(i) + 1 \tag{2.23}$$

where $0 \leq k < m$ (total number of SIFT features), $0 \leq i < n$ (total number of segments) and T is the variable which stores the total number of detected SIFT features in every segment. By calculation the maximum value of $T(i)$, i will index the label of the target object.

2.2.6 Result and conclusion

Fig. 2.25 presents experimental results of the proposed image processing algorithm for detecting different objects. In the first three pictures, the system has detected a cube in different poses. Pictures 4 and 5 illustrate the capability of the proposed algorithms to detect the complicated objects e.g. puncher. The last three pictures show how the proposed algorithms are able to detect the target object in different light conditions and even if the light is unequally distributed. The cycle time could range between 1.5 sec and 2.5 sec per frame, depending on SIFT features number and image size.

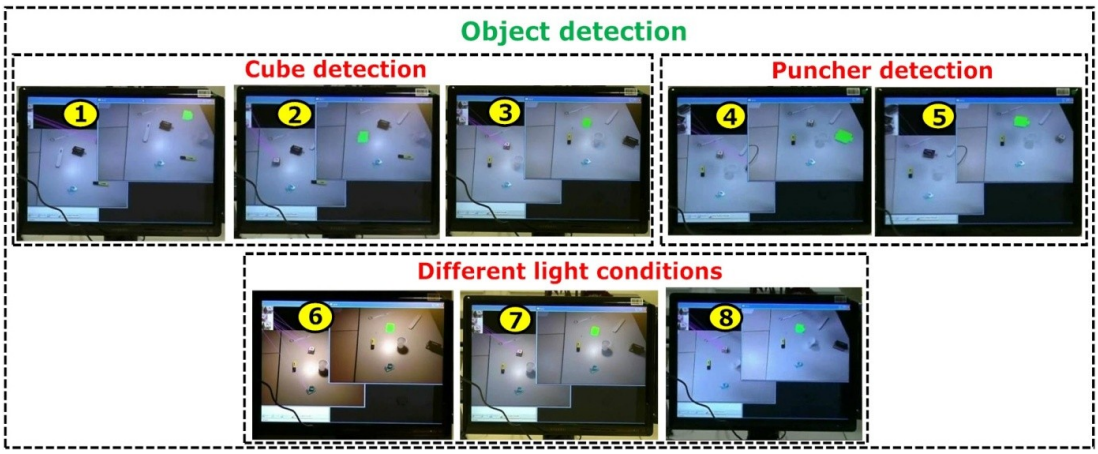

Fig. 2.25 Result of object detection

In conclusion, the previous section has presented the proposed vision algorithm which helps the robot system to detect and to segment complicated shape objects without any prior knowledge about their models. The proposed algorithm is able to segment the objects even if they have bad contours, especially when the illumination is permanently changing or even if the objects are located on conveyor or movable surface. The limitation of the proposed algorithms that the objects should be isolated from each other at least with one centimeter, i.e. the scene should not be cluttered. This proposed vision algorithms will be implemented later in the automated robot system to deliver model-free objects to the human hand.

2.3 Detection of object carried by human hand

Most of robot applications which require interaction with human intend to hand over different objects between both parties. In general, most of previous systems have supposed that the transfer task will be exclusively performed by human. However, how the task will be performed if the human is blind, elderly, disabled or concentrating on something else? In this case the transfer task should be exclusively controlled by robot, where the human could be considered as the weakest part. This section will present vision algorithms which help the robot to recognize and to segment any object carried by the human hand in order to hand it over from the human hand even if the human is disabled, blind etc. The proposed vision algorithms are able to detect and to segment any carried object with no idea about its model in a very short cycle time, even if it is textureless, if it has the same color as the human skin and if the light conditions are changed strongly. The proposed algorithms are supported by successful results on different kinds of object.

2.3.1 Introduction

The main goal of image segmentation is the detection of meaningful structures in a cluttered scene. Most current segmentation techniques depend on local image properties such as feature similarity (color, brightness, texture, motion etc.) to detect coherent units or objects. Unfortunately, image segmentation becomes very difficult in poor data conditions like shadows, occlusions and noise especially if the objects have the same features. In image processing, segmentation and recognition have reciprocal relationship. In other words, segmentation helps recognition and recognized objects constraint segmentation precisely.

Many papers have considered image segmentation as important pre-processing step for object recognition. Otherwise, other papers have proposed algorithms for object recognition in order to improve the segmentation of the objects, especially when the robot system will perform some tasks on the segmented object e.g. grasping, transferring etc. In the presented task, handing over unknown objects carried by human hand, the segmentation should be performed precisely (calculating graspability and ensuring the human safety). However, using object recognition before segmentation will not be useful in this application for the following reasons:

1. In general object recognition is divided into two main methods: Object recognition from 2D images and 3D object recognition. Usually, in the case of model-based recognition, the 3D model is constructed offline by acquiring the range image from multiple viewpoints of the object. The range image will be stored in a model library. During online recognition, the current range image of the scene will be compared with the models of the library database in order to recognize the target object. Model-based object recognition will be more complicated when the scenes are complex, especially in the presence of clutter (noise, object carried by human hand, etc.) and occlusions (multiple objects overlap).

2. The required time for the recognition and then segmentation of 2D cluttered and partial occluded object is not enough to perform real time visual servoing (grasping, handing over

etc), e.g. In (Main, Bennamoun, & Owens, 2006) the average time taken to recognize and segment a single object in a cluttered scene was about 2 minutes.

3. The recognition of a single object and determining its pose from a single image requires solving two problems: Finding enough correct correspondences between image features and model features, and estimating the model pose that best agrees with that set of correspondences.

4. For 2D recognition techniques, they are sensitive to illumination, shadows, scale, occlusions. An example is MOPED framework (Collet, Martinez, & Srinivasa, 2011) which integrates single-image, multi-image object recognition and pose estimation.

MOPED framework has used highly discriminative locally invariant features for matching such as SIFT (Lowe, 2004) or SURF (Bay, Ess, Tuytelaars, & Gool, 2008). Mismatched correspondences are inevitable by SIFT or SURF, so MOPED framework has combined them with some robust estimation techniques such as RANSAC (Fischler & Bolles, 1981). However, the recognition performance of MOPED is ultimately tied to the ability of finding enough local features in a given object, e.g. if an object is not textured enough, too far away, or it has large specular reflections on its surface, the feature extraction/matching steps might not find enough correspondences to perform any kind of recognition. Furthermore, the time factor will not be enough to perform real time visual servoing with all the previous algorithms. On the other side, many previous works have proposed algorithms to detect free open human hand and to analyse its gestures (Lambrecht & Krüger, 2012), (Chuang, Chen, Zhao, & Chen, 2011) and (Bergh, Carton, & Nijs, 2011). However, what about if the hand is loaded with unknown object, how the robot can detect the human hand, how the robot can segment the object from the hand and how the robot can track the carried object in the real time when human moves his/her hand randomly?

This work has proposed two algorithms which are working in different conditions: 1. Skin color based approach which combines RGB and Depth information. 2. Wrist model based approach which combines IR and Depth information. Using the proposed algorithms, the vision robot system will be able to detect and to segment any object carried by human hand even if the object has not enough SIFT or SURF features, if the object has complicated form (model-free object), if it has the same color of the human skin and even if the light conditions are changing strongly. The proposed vision algorithms could be used in different application, e.g. in service robotics, human robot teamwork etc.

2.3.2 Skin color based approach

The main goal of proposed algorithm is to segment the carried object from the human hand precisely and to define the boundary between them. Fig. 2.26 illustrates the main steps of the skin color based approach which is implemented by the language C and OpenCV library. As shown in Fig. 2.26, the first step of this approach is sending a request to read the data of the Kinect camera.

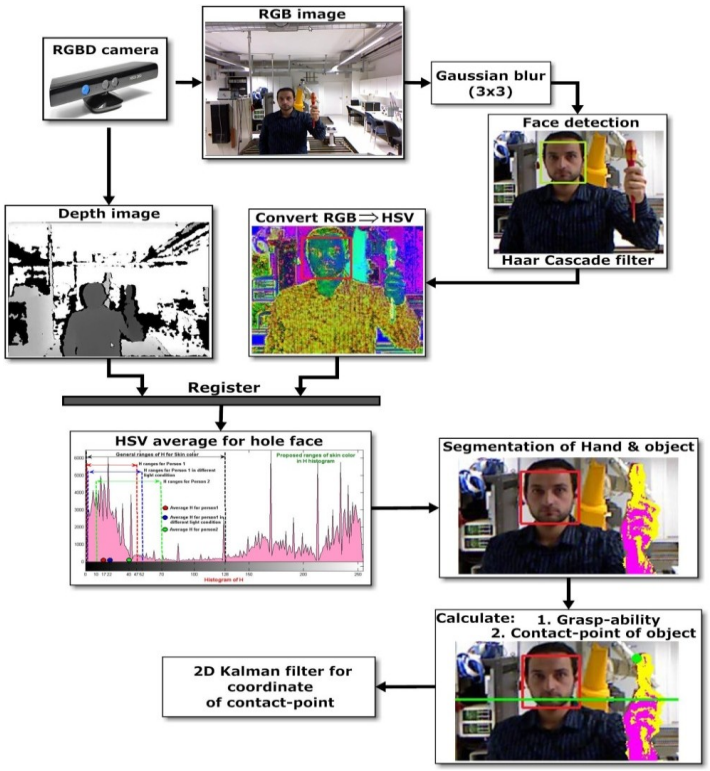

Fig. 2.26 main steps of skin color based approach

2.3.2.1 Kinect data

The open Kinect library system can collect data from camera in two modes: "call-back" synchronous mode and request-to-answer asynchronous mode. The optimal mode in the proposed algorithms is using request-to-answer mode, because the required time of image processing for one frame far exceeds of Kinect frame rate. In request-to-answer mode, the vision algorithm requests a new frame only if it completes the processing of the previous frame. The cycle time of request will not affect on the total cycle time because of its smallness. To reduce the noise of the images, two frames Img_1 and Img_2 will be captured and the average of them will be calculated and used as one frame as follows (R = red, G = green, B = blue):

$$R(i,j) = \frac{\left(R_{Img_1}(i,j) + R_{Img_2}(i,j)\right)}{2}$$
$$G(i,j) = \frac{\left(G_{Img_1}(i,j) + G_{Img_2}(i,j)\right)}{2} \qquad (2.24)$$
$$G(i,j) = \frac{(G_{Img_1}(i,j) + G_{Img_2}(i,j))}{2}$$

where i=1,…,640 ; j=1,…,480. Decreasing the frame rate of the camera will not affect the total cycle time of the image processing. The skin color based approach is able to process 5

frames/second. Therefore, a good quality of source data will lead to a successful image processing and will reduce the risk of unsuccessful operations.

Next step as shown in Fig. 2.26 is smoothing the frames using Gaussian blur (Nixon & Aguado, 2008) with aperture size 1. The general equation for Gaussian blur can be described as follows:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.25}$$

where x is the distance from the origin in the horizontal axis, y is the distance from the origin in the vertical axis, and σ is the standard deviation of the Gaussian distribution.

2.3.2.2 Face detection

The next stage is face detection by using OpenCV cvHaarDetectObjects() function. In openCV, predefined Haar-cascades face database has been used. At the end of this step, a rectangle will be drawn around detected faces FaceReg(i) as shown in Fig. 2.26, where i = 1,…,N (N is the number of the detected faces in the frame). The width and length of rectangle are (FR_{Wid}(i), FR_{Len}(i)). In the next step, the vision system will define the nearest face to the camera (in case of more than one person) by calculating the depth information of the middle point of FaceReg as follows:

$$Face_z = \min\left[Z\big(FR_x(i) + FR_{Wid}(i)/2\,, FR_y(i) + FR_{len}(i)/2\big)\right] \tag{2.26}$$

where $(FR_x(i), FR_y(i))$ are the coordinates of the left-upper point of the face rectangle. Z(x,y) is the depth matrix of the coordinate x and y.

All the pixels of the nearest detected face are included inside the face rectangle as matrix $\{Face_x, Face_y\}$. If no person is detected in current frame, the status will be “NO FACES” and the frame will be ignored.

2.3.2.3 Convert RGB->HSV

Next step converts the image from RGB to HSV color space, because HSV space is more related to human color perception. HSV stands for hue, saturation and value, it is also often called HSB (B for brightness). CvCvtCopor() and CvCvtPicToPlane() functions in openCV have been used.

Fig. 2.27 HSV image

2.3.2.4 Calculation the HSV averages of the detected face

A previous work (Phung, Bouzerdoum, & Chai, 2005) has shown that human skin color locates in a wide range of H and S spaces. By assuming that the range of HSV space is (0,...,255), the general range of human skin color will be $G_{H_{min}}$=0, $G_{H_{max}}$=128 and $G_{S_{min}}$=59, $G_{S_{max}}$=175 and there is no general range for V space. It is clear that the general range of human skin color in HSV space is very wide, e.g. V values could spread in whole space, because V values are related to light condition and the quality of streams. H and S values could spread almost in the half of the space. Hence, the general ranges could be sufficient to detect skin but they will not be enough to segment the hand from the object in complicated scene or in different light conditions, especially if the object has almost the same color of the skin such as wood objects.

In the proposed algorithm, face detection will not be used only to define the safety zone but it will also be used to define the average skin color of the active person in HSV space. Active person is the person who will interact with the robot and his/her face is the nearest face to the robot (if more than one person is in the frame). Hence, the color of human skin will be updated in every frame if light source, active person and skin reflection are changed. Therefore, this will narrow the ranges of skin color in HSV space and define exactly the color of the human hand, even if the human has shining white or dark skin color.

The calculation of the average HSV values of the detected face will be as follows:

$$if\,(Face_z - 40 < Z(i,j) < Face_z + 40)\ then$$

$$\{Average_H = \left(\sum_{i=FR_x}^{FR_x+FR_{Wid}} \sum_{j=FR_y}^{FR_y+FR_{len}} H(i,j) \right) \Big/ N\} \qquad (2.27)$$

The “if” condition ensures that all pixels of the face rectangle locate on the human face with depth deviation less than ±40mm. N is the number of pixels of the rectangle which verify the depth condition. For better optimization of $Average_H$, sampling pixels could be later collected only from forehead or nose skin. In the same way $Average_S$ and $Average_V$ will be calculated.

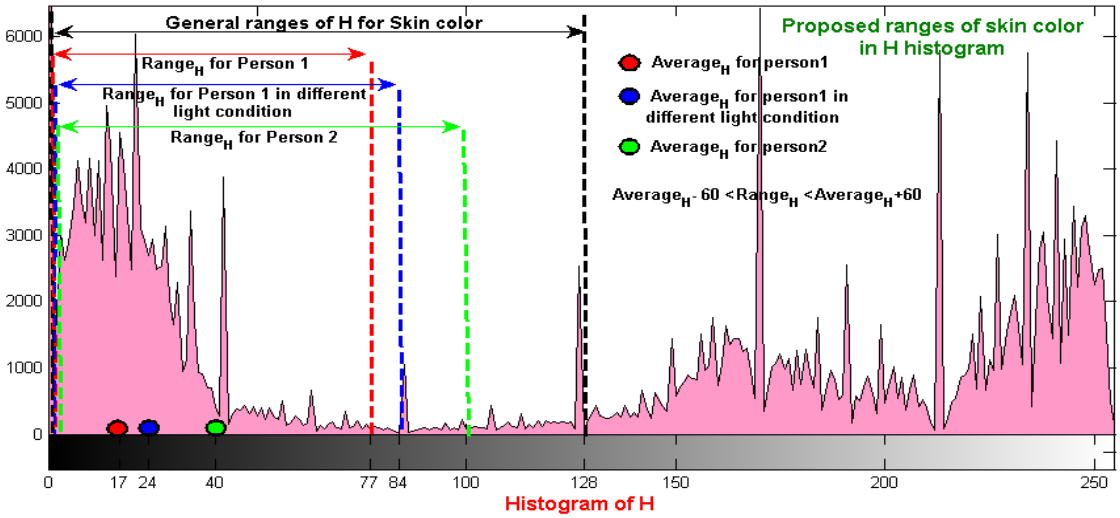

Fig. 2.28 Histogram of H

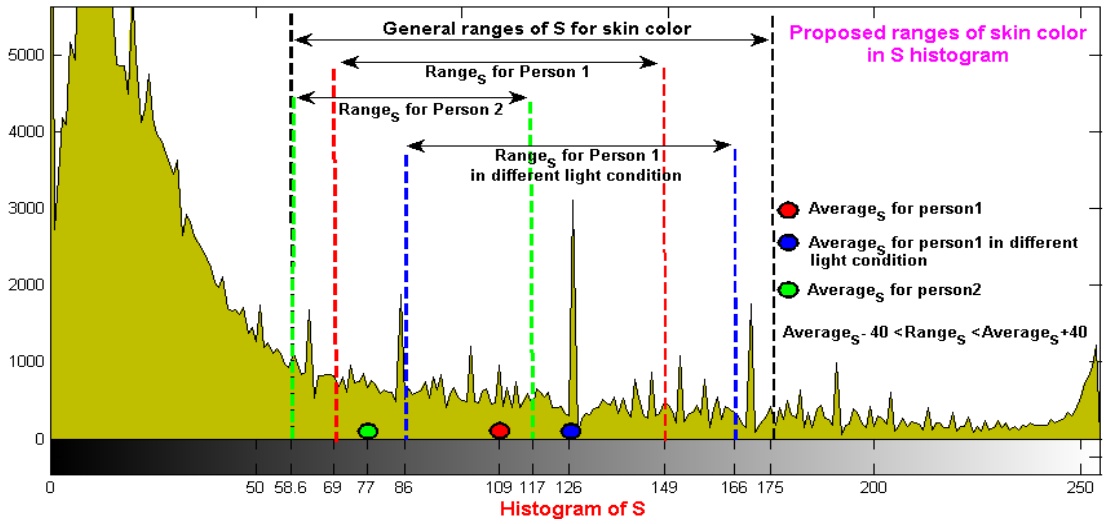

Fig. 2.29 Histogram of S

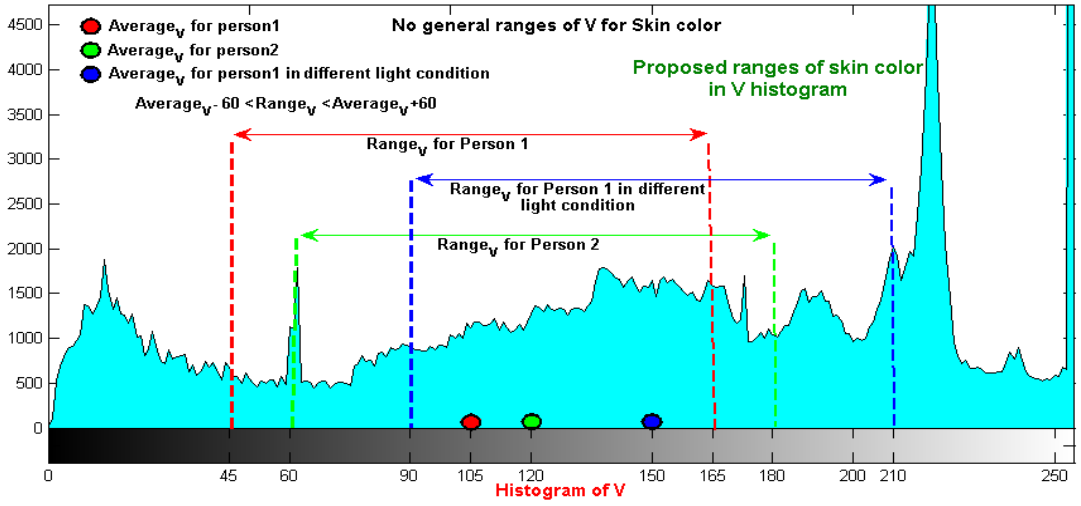

Fig. 2.30 Histogram of V

Fig. 2.28,Fig. 2.29 and Fig. 2.30 illustrate the advantages of the proposed algorithm to detect the human skin color depending on face color by comparing with the general ranges of the human skin color even under different light conditions or with different people. Fig. 2.28,Fig. 2.29 and Fig. 2.30 present the histograms of HSV components in different light conditions and with different people. The black lines present the general ranges of the HSV of the human skin color. As is shown, the general ranges $(G_{H_{min}}, G_{H_{max}})$ and $(G_{S_{min}}, G_{S_{max}})$ are always constant and they have a wide range. Going out of experiments, in approach proposed by us the ranges of the skin color in HSV color space will be as follow:

$$\begin{aligned} Range_H &= Average_H \pm 60 \\ Range_S &= Average_S \pm 40 \\ Range_V &= Average_V \pm 60 \end{aligned} \tag{2.28}$$

The proposed values of $Range_H$, $Range_S$ and $Range_V$ allow us to segment the human hand from the object precisely (even if it has a color near to the human skin). Furthermore, if the object has exactly the same color as the human skin such as wood objects, the proposed algorithm is able to segment them, if the human rotates his/her hand a little. During the rotation, a huge difference in the HSV values will appear between the light reflection from human hand and the light reflection from the object. The human skin doesn't reflect the light, therefore during the rotation the object color will be outside of the HSV ranges of the human skin, even if the object has the same color of the human skin.

In Fig. 2.28, Fig. 2.29 and Fig. 2.30, the experiments are repeated three times: First two times they are performed on the same person but with different light condition (red and blue points). As is shown there is big difference between the values of $Average_V$ with the light changed, because the V component is directly related to the light condition. The third experiment is performed on different person whose skin color is different from the first person. As shown, using the proposed approach to define the skin color is more exact. Because the skin color ranges will always be updated in every frame, then if any changes in person or light conditions have occurred it will directly effect the range values. If one value of the proposed ranges exceeds the max. or min. value of the general range, the value of the general range value will be taken, e.g. $Range_S$ of the person2 in Fig. 2.29.

2.3.2.5 Detection and segmentation of hand and object

The detection and segmentation phase of the human hand and the object consists of the following steps:

- Segment human body and object from the background.
- Define the area of interest (which contain the human hand and the object)
- Segment the human hand from the carried object.

First step will segment the human body and the object from the background. The system will search for all connected pixels starting from the middle point of the rectangle in the depth space. All the connected pixels of the human body will be distinguished from the background as is shown in Fig. 2.31 by using CVfloodfill() function. As shown in Fig. 2.31, this step will segment the object as a part of the human body (connected with human hand).

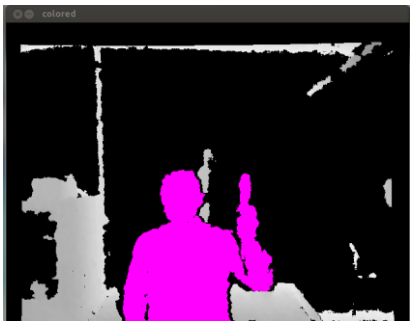

Fig. 2.31 Background and Body segmentation

Second step defines the area of interest which contains only the active hand and the object. For reason of safety, the active hand, which carries the object, should be far from face rectangle with minimum distance of 80mm, else the task will be cancelled. The area of interest (blue rectangle in Fig. 2.32) consists of all pixels which are segmented as human body (magenta color in Fig. 2.31) and which are further than 80mm from the middle point of the human face.

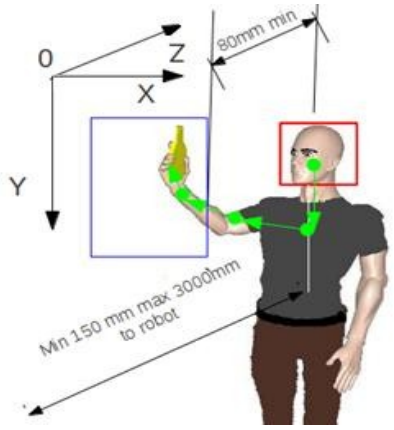

Fig. 2.32 Area of interest

Third step defines the skin color of the human hand and the boundaries between human hand and the target object. Mask M[x,y] will cover the whole frame. Depending on the mask values, the segmentation between human hand and the object will be performed. The values of the mask M[x,y] will be defined by combining the general range of human skin color and the proposed range of skin color as follows: If a pixel (x,y) is out of area of interest or it is not connected with the middle point of the face in the depth space, the values of the mask will be:

$$M(x,y) = 0 \tag{2.29}$$

On the other hand, if the pixel (x,y) is inside the area of interest and it is connected with the middle point of the face, there will be two color conditions:

First condition: (general condition)

$$if\ \left(G_{H_{min}} < H(x,y) < G_{H_{max}} \ \ and \ \ G_{S_{min}} < S(x,y) < G_{S_{max}} \right) \text{ then}$$

Second condition: (proposed conditions)

$$if\ \left(|H(x,y)| < |Range_H| \ \ and \ \ |S(x,y)| < |Range_S| \ \ and \ \ |V(x,y)| < |Range_V| \right) \text{ then}$$

If both conditions are fulfilled, the pixel has color which is within the range of human skin:

$$M(x,y) = 1 \tag{2.30}$$

Otherwise, the pixel will belong to the object:

$$M(x,y) = 2 \tag{2.31}$$

Depending on the values of the mask $M(i,j)$, where i = 1,...,640 and j = 1,...,480, the pixels will be colored as yellow (object) or magenta (hand) as shown in Fig. 2.33. The calculation of the graspability and tracking point will be illustrated later in chapter 4 using the current results of image processing.

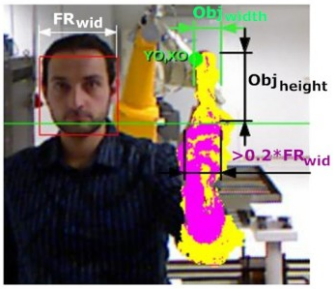

Fig. 2.33 segmentation of human hand and object

Let's define a new mask $M_{hand}(i,j)$ as follows:

$$M_{hand}(i,j) = \begin{cases} 0 & if \;\; M(i,j) = 2 \\ 1 & if \;\; M(i,j) \neq 2 \end{cases} \qquad (2.32)$$

Depending on the human hand mask, the system will define the boundary line (green line) between the human hand and the object by scanning every row in the frame from top to bottom.

To avoid any communication error or when the depth information is broken, a protection equation will be written as follows:

$$\sum_{i=0}^{640} \sum_{j=0}^{480} \widehat{M}(i,j) < 0.25 * 640 * 480 \qquad (2.33)$$

Equation (2.33) tests if the area of interest (which contains the hand and the object) is less than quarter of the image area, otherwise the depth information or communication between the camera and the computer is broken.

Fig. 2.34 presents experimental results of the skin color based algorithm on different objects. As shown, most of the objects are textureless (SIFT and SURF cannot be used) such as sponge(2) or cube(8), some objects have almost the same color as human skin, such as wood-cube(8), ruler(7), hammer (8) or wood-meter(5). Most of the objects have different models and some of them have complicated form such as screwdriver (6), screwdriver(9) or red box(4). Hence, we can conclude that:

- Skin color based approach takes opportunity to define exactly the contours of hand and object (even for objects with the same color as the human skin, if the human rotates his/her hand).
- Skin color based approach doesn't depend on any model of the human hand or object.

- Skin color based approach uses RGB camera information as basic data and depth image as additional data. Because RGB data are less noisy than depth data the method shows good capability.
- Skin color based approach is able to process at least 5 frames per second, which gives us opportunity for real time visual tracking.

Fig. 2.34 Results of different objects detection

There are some limits of this approach:

- The capability of segmentation depends strongly on object's color and light conditions.
- If the object has the same color as the human skin and the human doesn't rotate his hand, the segmentation could be performed without success.
- This approach can't be implemented if there is no sufficient light source or where some special kind of light suddenly switched (flashing, colored lamps etc).
- This approach can't be implemented if the human wears gloves or has Vitiligo disease.

2.3.3 Wrist model based approach

As is mentioned previously, the proposed skin color based approach cannot be used if the illumination is very low or the light source has different color temperatures, if the human has Vitiligo disease or wears gloves or if the object color has the same color of the human skin, and the human doesn't rotate his hand. Hence, in this work a second approach will be proposed which depends on the IR pictures of the Kinect camera. IR frames will give us the opportunity to achieve the transport of objects between human hand and the robot even if the work room is very dark or light conditions are changing or even when the human wears gloves or has Vitiligo disease, because the wrist model based approach will not depend on the color.

The infrared images are captured with illumination by embedded camera projector and they consist of special net of patterns. These patterns are used to estimate the depth information. However, the existence of these patterns in the IR frame will be considered as an impulse noise. Therefore, these patterns will be removed by using the median filtering (Hwang &

Haddad, 1995). The general equation for multidimensional vector based median filtering can be written as follows:

$$y_{m,n} = \left\{ x_k : \sum_{j=1}^{N} |x_j - x_i| \geq \sum_{j=1}^{N} |x_j - x_k| \ , \forall\, i \in [1, \dots, N] \right\} \tag{2.34}$$

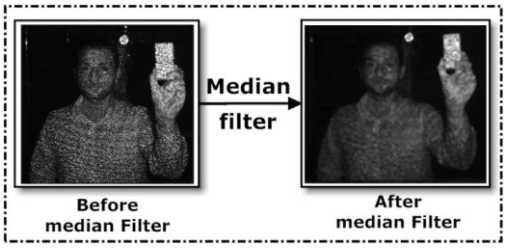

Fig. 2.35 Median filter

Fig. 2.35 presents the image after using median filter. Infrared captures take a sketchy picture of the objects which locates in distance 0-3000mm from the camera. The obtained details of the IR frames give us opportunity to separate human's body from the background and also for face detection. However, these details are not enough to segment between human hand and the object based only the IR frame. Because of that the depth information will be combined with the IR information in order to segment between the human hand and the object.

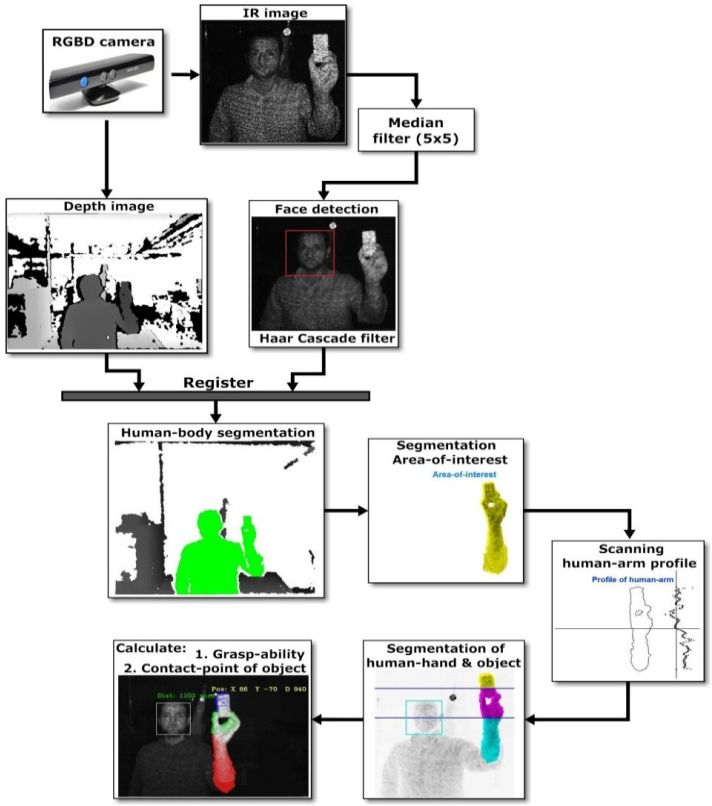

Fig. 2.36 Main steps of Wrist model based approach

Fig. 2.36 illustrates the main steps of the proposed wrist model based approach to segment any objects carried by human hand. Using IR frame, the system will be able to detect the human face by openCV, cvHaarDetectObjects() function as explained previous approach.

2.3.3.1 Human body segmentation

Next step will start by segment the human body and the object from the background. The system will search for all connected pixels starting from the middle point of the face rectangle in the depth space. All the connected pixel of the human body will be distinguished from the background as is explained in the previous approach. This step of segmentation will segment the object as a part of the human body, because the object is also connected to the human body (connected with human hand). As a result, as mask $H(x,y)$ for the human body and all connected pixels will be calculated, where $H(x,y) = 1$ when the pixel is belong to the human body or to any object connected with the human body, otherwise $H(x,y) = 0$. In Fig. 2.37, the human body and the carried object are shown in the green color.

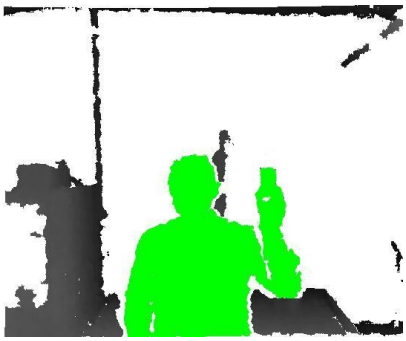

Fig. 2.37 Segmentation of the human body

2.3.3.2 Area of interest

In the next step, the system will define the mask of area of interest $AOI(x,y)$ (as shown in Fig. 2.38) which contains the active hand and the object. The active hand is the hand which is carrying the object. The segmentation of the area of interest will be as follows:

1. Finding the nearest point of the mask $H(x,y)$ to the robot.

$$z_{min} = \min_{z} \{ z[\, H(x,y)\,] \}, \qquad x = 0, \dots, 640;\ y = 0, \dots, 480 \tag{2.35}$$

2. Next step will define the depth map of the area of interest which is the located area between z_{min} and $z_{min} + 150mm$.

$$AOI(x,y) = \begin{Bmatrix} H(x,y), & if\ \ z_{min} \le Z[\, H(x,y)\,] \le z_{min} + 150 \\ 0\ , & otherwise \end{Bmatrix} \tag{2.36}$$

The depth map of the area of interest will be:

$$AOI_z = z\,[\, AOI(x,y)\,], \qquad x = 0, \dots, 640;\ y = 0, \dots, 480 \tag{2.37}$$

Where AOI_z is the depth matrix which contains the depth points of all pixels which belong to the human hand and to the object. The segmentation of the human hand and the object will be performed depending on the human hand profile.

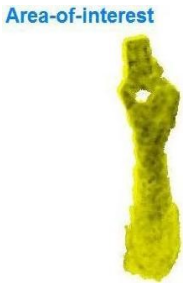

Fig. 2.38 Area of interest

2.3.3.3 Analyzing of wrist profile

Reference (Smisek, Jancosek, & Pajdla, 2011) has presented the depth resolution as a function of the distance in Kinect camera. The size of the quantization step q, which is the distance between the two consecutive recorded values, was found as function of the depth z as follows:

$$q(z) = 2.73 \cdot z^2 + 0.74 \cdot z - 0.58[mm] \qquad (2.38)$$

where z is the depth values in [mm]. Hence, by assuming the work space (distance between human hand and the robot) will range between 0.5m-3m, the values of q will range from $q(0.5m) = 0.65mm$ to $q(3m) = 26.21mm$. Accordingly, the optimal distance between the human hand and the robot ranges between 0.5m until 2m in order to analyze the profile of human hand, when the Kinect camera is used. By using another vision device which has better depth resolution, the proposed approach could be used in longer distances.

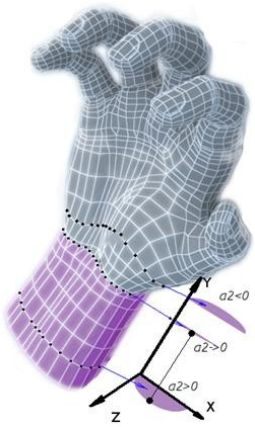

Fig. 2.39 Human hand profile

Fig. 2.39 presents the basic principle of the proposed approach. The most important region of the hand is the wrist (the joint between the hand and the arm). By analyzing the cross section of the human hand parallel to the plane xz, it can be concluded that a second order algebraical

equation can approximately represent all the points of cross section at any value of y in the wrist region as follows:

$z(x_i, y = k)$	0_0	...	0_M	$z(x_{M+1}, k)$	$z(x_{M+2}, k)$	...	$z(x_{M+N}, k)$	...	0_{640}
$x_i, i = 0, \dots, 640$				x_{M+1}	x_{M+2}	...	x_{M+N}		

Table 2.6 z direction as function for x in one cross section

where k is the step of the cross section in the direction of y axis, M represents the index of the first pixel in the area of interest and N is the number of pixels in the cross section.

$$\hat{z}_k(x_i) = a_2 \cdot {x_i}^2 + a_1 \cdot x_i + a_0; \quad i = 1, \dots, N \tag{2.39}$$

Table 2.6 illustrates the function z of x. The depth information of all x pixels which don't belong to the human hand will be zero. Using least square second order polynomial approximation, the system can calculate the coefficients a_2, a_1 and a_0 of equation (2.39) to be approximate solution of all points of row k from the following criterion:

$$Q = \sum_{i=1}^{N} (z_k(x_i) - \hat{z}_k(x_i))^2 \tag{2.40}$$

where $z_k(x_i)$ is the real depth values and $\hat{z}_k(x_i)$ is the estimated depth values. By obtaining the partial derivatives of Q with respect to the coefficients (a_2, a_1, a_0) and equating these derivatives of zero, the system will get three equations with three unknown variables (a_2, a_1, a_0). By solving these derivatives equations, the system will calculate the values of the coefficients (a_2, a_1, a_0) in (2.39). In general, the coefficient a_2 in second order equation describes the flexion of the curve. Hence, the coefficient a_2 in equation (2.39) will describe the flexion of the approximated cross section of human hand set points at step k. By scanning all the cross sections of the human hand, especially in wrist region, it can be concluded the following:

- If the cross section locates below the wrist joint, the coefficient a_2 will always be positive. (as shown in Fig. 2.39, the region of violet color)
- The value of coefficient a_2 will become unstable (sometimes negative and sometimes positive) exactly when the cross section crosses the wrist joint upwards.

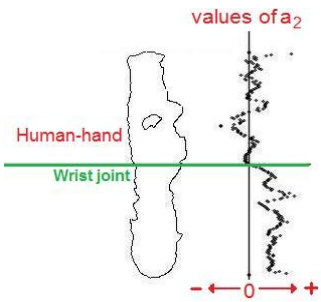

Fig. 2.40 Values of coefficient a_2

Fig. 2.40 shows the changes of the value of coefficient a_2. When the cross section is below the wrist joint the values of coefficient a_2 is always positive. Whereas, when the cross section is above the wrist joint, the values of coefficient a_2 will range between positive and negative values. To summarize, the first zero-crossing of the values of coefficient a_2 will indicate to the position of the wrist joint of the human hand (when the scanning is performed from elbow toward the hand). The proposed approach will give us the opportunity to estimate the position of the wrist joint.

According to the hand/face size ratios (Koscinski, 2011), the length of the hand is related to the length of the face. After defining the position of wrist joint, the safety distance for human hand $\mathrm{SD} = 0.8 * FR_{len}$ will be segmented as human hand. The last part of the area of interest will be segmented as a target object, as shown in Fig. 2.41.

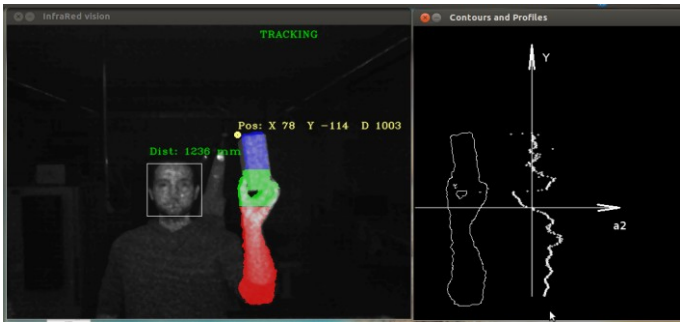

Fig. 2.41 Human hand and object segmentation

The last phase of this approach defines the first contact point of the object and then calculates the graspability, e.g. if the robot gripper can grasp the object or not, (see Chapter 4).

Fig. 2.42 presents experimental results of the proposed algorithm on different objects. The red region represents the human part of the area of interest which locates below the wrist joint. The green region is the human hand (according to the ratio between the length between human hand and human face) and the blue region represents the mask of the object. The proposed approach has proved its ability to segment between the human hand and objects, even if the object has the same color as the human hand such as sponge(4), cube(1), hammer(3) or wood-meter(5) and even if the object has different models such as screwdriver(7), hammer(3) or wrench(2).

To conclude:

- Wrist model based approach can be implemented even if the human wears gloves or has Vitiligo disease.
- Wrist model based approach is able to work in different light conditions starting from complete darkness and ending with different color-temperature lamps.
- Wrist model based approach is able to segment the human hand and object even if they have the same color.

- Wrist model based approach has very fast cycle time of about 10 frames per second, which give opportunity to perform a very fast real time visual tracking.

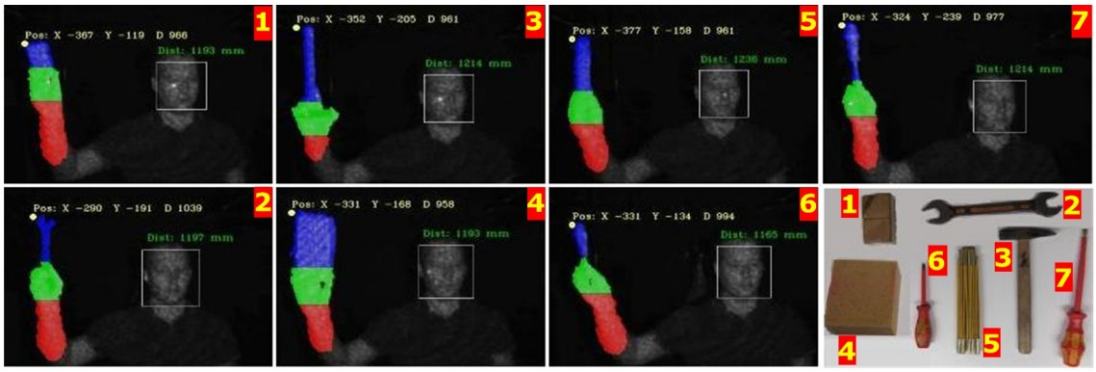

Fig. 2.42 Wrist model based approach on different objects

The limitations of this approach are the following:

- In Wrist model based approach, the medial side of the human hand (where the wrist joint clearly appears) should face the camera or be rotated with maximum angle of $\pm 45^{\circ}$.
- If the lateral side of the human hand faces the camera, the profile of the human hand will be flat and this approach couldn't be successfully performed. However, this position isn't fit for handing over an object. When the lateral side of the giver hand faces the receiver, this means that the fingers of the giver and the objects are directed toward the giver itself not toward the receiver, which is bad position for handing-over an object.

On the whole, the best possibility to segment any object from the human hand without any idea about the model of the object can be performed by combining both skin color based and wrist model based approaches. However, the present Kinect technology doesn't offer this possibility (one should switch between the two modes). The future work could go in direction of implementation both approaches as one approach in a vision device which can deliver color and infrared frames at the same time.

Fig. 2.43 Dataset of different segmented objects

The proposed algorithms have been repeated for more than 15 different objects carried by different users. Fig. 2.43 shows the dataset of segmented objects using the proposed algorithms and Table 2.7 presents the cycle time of them.

ID	Object	Skin color based approach	Wrist model based approach
1	Marker	185ms	93ms
2	Screwdriver (1)	189ms	96ms
3	Hammer	188ms *	94ms
4	Complex shaped box	186ms	98ms
5	Meter (1)	184ms *	96ms
6	Screwdriver (2)	183ms	94ms
7	Allen wrench	184ms	93ms
8	Screwdriver (3)	184ms	94ms
9	Meter (2)	189ms *	97ms
10	Sponge (1)	193ms	98ms
11	Couple of cubes	185ms *	92ms
12	Plastic cup	188ms	94ms
13	Sponge (2)	190ms *	96ms
14	Milk bottle	194ms	98ms
15	Wrench	189ms	94ms
16	Puncher	188ms	95ms
17	Stapler	185ms	93ms
18	Cube of letter A	183ms	92ms

Table 2.7 Cycle time of objects segmentation

* These objects have similar color of skin color, so the skin color based algorithm works well if the human rotates his/her hand

2.3.4 Conclusion

To summarize, this chapter has proposed three real time image processing algorithms for detecting different kind of model-free objects. The first algorithm is able to detect objects which have simple geometric shapes even if they are stuck together. This algorithm will be implemented for detecting books in the library automation scenario. The second one detects any object locates on flat surface, e.g. conveyor, table etc, even if the object is complicated and has complex structures. The last algorithm detects any object carried by human hand even if it has the same color as human skin color. The last two algorithms will be implemented in scenario of handing-over objects from/to human hand.

Chapter 3

Visual servoing

Visual servoing is novel approach to control robots in the tasks which are defined visually. In other words, the measurements for feedback are obtained from the vision system in order to control the motion of the robot in the real time. In general, visual feedback is used mainly for in gross motions. When the robot is far from an object the visual servoing is adopted, the relative pose of the robot with respect to the object and the image features of the object could be calculated using vision processing. Visual system may consist of one, two or more cameras. If more cameras are used to observe the same object of a scene, it is possible to retrieve information about its depth by evaluating its distance with respect to the visual system (3D vision or stereo vision). Actually, vision information can be used as sensory input in open-loop as well as in the closed-loop. However, the visual servoing approach is performed only inside the closed-loop robot control, because in the open-loop the vision sensor will represent the initial extraction of the features to generate directly the robot motion sequence and these features and motion could be off-line generated. On the contrast, closed-loop robot system uses the vision as real time sensor and it consists of two phases: tracking and control. Tracking provides a continuous estimation and update of features during the robot/object motion. Based on this information, a real time control loop will be generated.

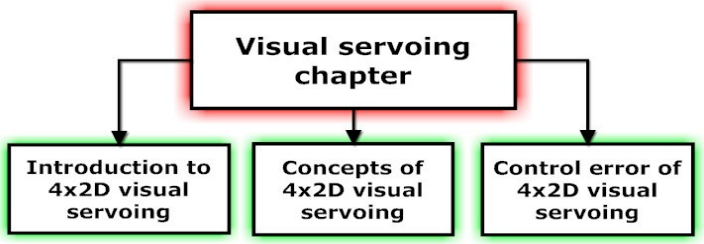

Fig. 3.1 Overview of Chapter 3: Visual servoing

The main contribution of this chapter is a proposed visual servoing approach which will benefit from the images which are obtained by Kinect camera (RGB-D camera). The proposed visual servoing approach is called 4x2D visual servoing which combines the correspondent color and depth images to build two new images. Using these 4 images the control error signals will be calculated in order to track the objects. Firstly, this chapter will introduce the 4x2D visual

servoing approach and the visible side coordinate system, after that it will illustrate the concept of the proposed approach and how the error signal will be calculated.

3.1 Introduction to 4x2D visual Servoing approach

All the previous visual servoing approaches (see Chapter 1) are proposed with consideration of using 2D camera, e.g.: 1. A single 2D camera will be used in case of IBVS. 2. In the case of PBVS stereo 2D camera system or single 2D camera with multiple viewpoints or multiple focusing factors could be used. In the last years the RGB-D cameras (such as Kinect camera) are spread widely everywhere. RGB-D cameras deliver depth and color images simultaneously and give us the opportunity to use this information in the visual servoing tasks. To exploit all the capabilities and advantages of the RGB-D cameras, this work will suggest a new approach of visual servoing. This approach will be called 4x2D visual servoing. 4X2D visual servoing combines the correspondence of color and depth images to build (u, w) and (w, v) images. By using these 4 images (RGB, depth, (u, w) and (w, v)) the system can calculate the control error signals at every subspace directly from the obtained images.

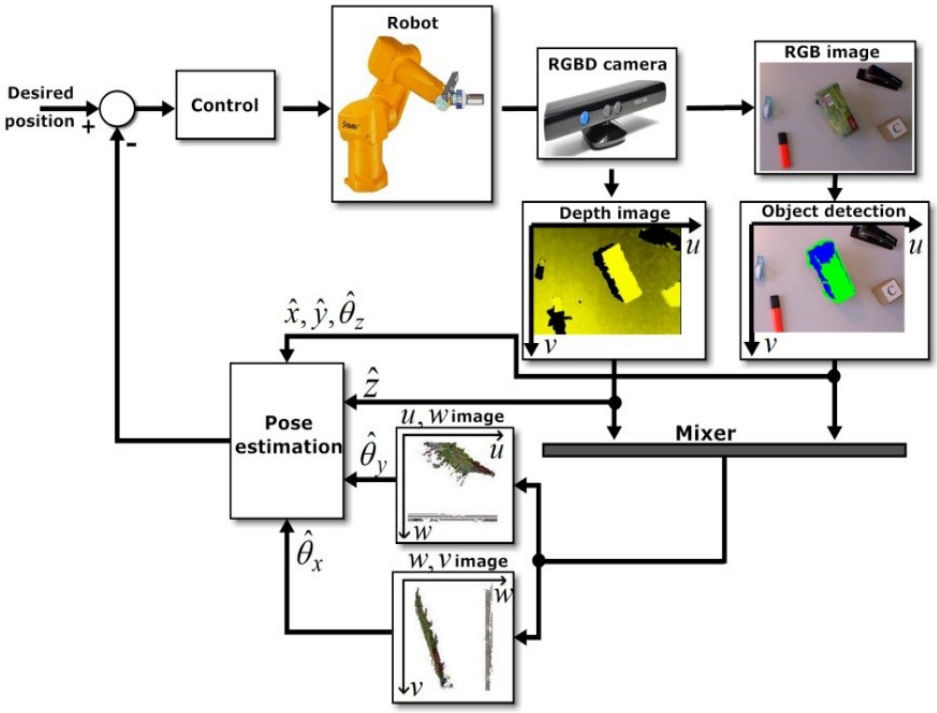

Fig. 3.2 4x2D visual servoing approach

In PBVS approach many methods are used to estimate the 3D pose of the object from 2D images such as stereo vision, photometric stereo, shape from shadow etc. These approaches have many problems and disadvantages, for example the stereo vision has two main problems 1. Correspondence problem: determining which pixel on the left image corresponds to which pixel on the right image. 2. Reconstruction problem: define a number of correspondence pairs to find location and 3D structure of the target object. Furthermore, the using of IBVS approach

in the 3D visual servoing applications is limited. Otherwise 4x2D visual servoing approach is a simple approach which allows to use IBVS in 3D space and to estimate the pose of the target object when PBVS is needed. Next sections will discuss some suggested concepts and notations to perform the 4X2D visual servoing approach, such as mixing of RGB/depth images, visible side coordinate system, projections in the 3D image space etc.

3.1.1 RGB/depth images mixing

As is known, Kinect camera can work in two different modes IR/depth or RGB/depth. The calibration error between IR image and depth image is zero because both images are coming from the same sensor. In the RGB/depth mode there is small translation between RGB image and depth image. This translation could be compensated by using a function in openNI library (see chapter 5). Hence, if the 3D information is needed, the calibration error between images is eliminated to zero.

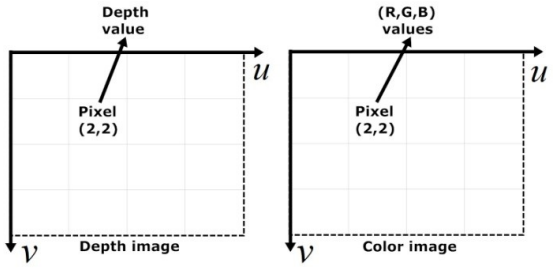

Fig. 3.3 Depth and color images

The depth image represents topographic view of the scene (matrix of pixels and each pixel contains a value representing the distance of the object point from the sensor). The color image is a standard output of a 2D digital camera which is used for texture detection. To produce more accurate sensory information Kinect performs a process called registration. The registration process's resulting images are pixel-aligned, which means that every pixel in the color image is aligned to a pixel in the depth image. In Fig. 3.3, the pixel(2,2) in the color image and the pixel(2,2) in the depth image are aligned, first one contains the values of (red, green, blue) and the second one contains the depth value of the same scene view.

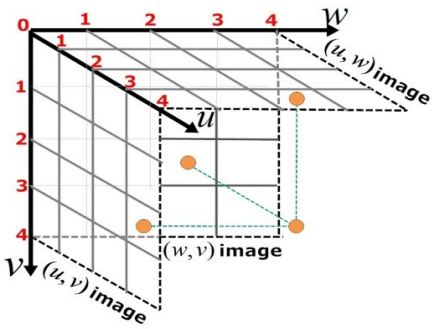

Fig. 3.4 3D space of images

By combining both color and depth images, we can generate what we will call 3D space of images as shown in Fig. 3.4. The 3D space of images is the 3D image coordinate system which contains three images (u, v), (u, w) and (w, v). The firs image (u, v) is the color image without any change. The two others (u, w) and (w, v) are new reconstructed images which are calculated by combining depth and color information. 3D space of images will give us information about the view of the scene in different projection planes.

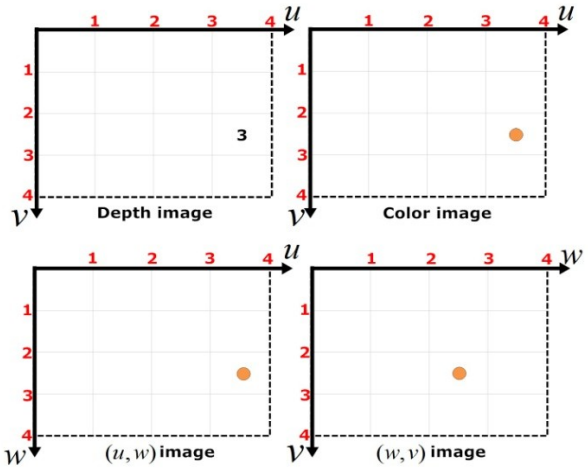

Fig. 3.5 example of (u, w) and (w, v) images calculation

Fig. 3.5 shows a simple example of calculation of one pixel in (u, w) and (w, v) images. By assuming that the Kinect camera has delivered the depth and color images while the value of the pixel(4,3) in the color image is e.g. orange (239,126,44) and the depth value of the same pixel is 3. This means the pixel(4,3) in (u, w)image and the pixel(3,3) in (w, v)image will have the same color (239,126,44).

3.1.2 Coordinate systems for the suggested approach

Fig. 3.6 shows three different coordinate systems which are camera (C), tool (T) and object (O) coordinate systems. The (T) coordinate frame is attached to the tool of the robot which could be 2 or 3 fingers gripper. There are two other coordinate systems, which are world coordinate system (W) and a new suggested coordinate system (S). The (S) coordinate frame is attached to the visible side of the target object which will be illustrated in the next section.

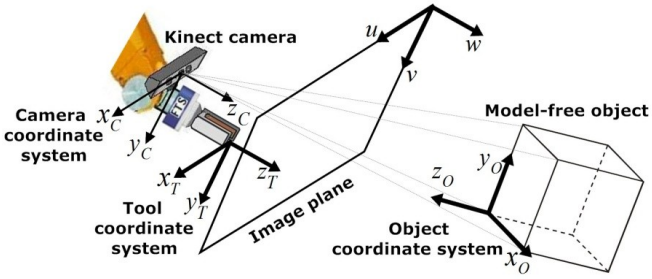

Fig. 3.6 Coordinate systems

3.1.2.1 A plane in three dimensional space

For better understanding of visible side coordinate system definition, this section will discuss some proprieties of a plane. In general, any plane in a three-dimensional space could be defined by specifying one point and the normal vector of this plane as shown in Fig. 3.7.

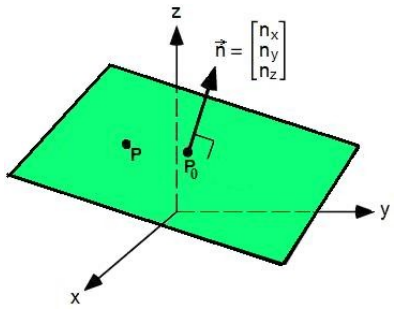

Fig. 3.7 A plane in three-dimensional space

If we assume that $\vec{r}_0$ is the position of one known point $P_0 = (x_0, y_0, z_0)$ in the plane and $\vec{n}$ is the normal vector of the plane, any point $P = (x, y, z)$ with position vector $\vec{r}$ will be in the plane if and only if the vector from P_0 to P is perpendicular to normal vector $\vec{n}$.

$$\vec{n} \cdot (\vec{r} - \vec{r}_0) = 0 \tag{3.1}$$

With (3.1) expanded:

$$n_x(x - x_0) + n_y(y - y_0) + n_z(z - z_0) = 0 \tag{3.2}$$

This is familiar equation of the plane:

$$ax + by + cz + d = 0 \tag{3.3}$$

The cross product of two vectors $\vec{a} \times \vec{b}$ is a perpendicular vector to both vectors $\vec{a}$ and $\vec{b}$. Hence, if we assume that both vectors belong to the plane, equation (3.1) could be rewritten as follows:

$$(\vec{a} \times \vec{b}) \cdot (\vec{r} - \vec{r}_0) = 0 \tag{3.4}$$

where

$$\vec{n} = \vec{a} \times \vec{b} \tag{3.5}$$

To generalize this concept, we will suggest a new coordinate system. This coordinate system will be related to major axis of the visible side (plane) of the object. In this work, the object's side (plane) which is seen by the camera will be called the visible side. The new coordinate system of object will be referenced by S (from side), so its axes of will be (x_S, y_S, z_S)

3.1.2.2 Visible side coordinate system

Visible side coordinate system is a new proposed coordinate system which is related only to the visible side of the target object (object's side which is seen by the camera). This section will

discuss the proposed coordinate system in detail starting from special cases until generalization. The visible side of the objects could be either a plane or a curved surface.

1. *Visible side as plane*

In this case, the axis y_S will be the parallel axis to the direction of the major axis of the visible side, the axis x_S will be parallel to the direction of the minor axis of the visible side of the object and the axis z_S will be perpendicular to the visible side plane (x_S, y_S) according to the right-hand rule, as shown in Fig. 3.8.

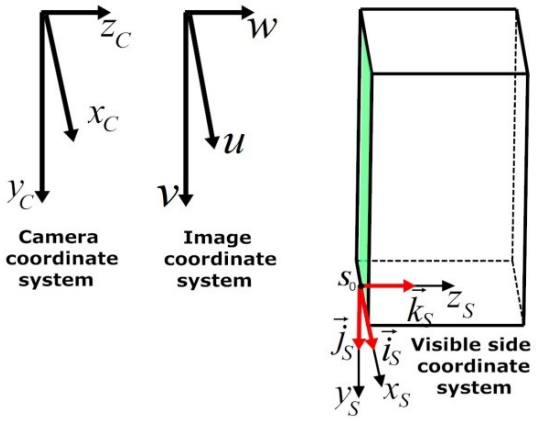

Fig. 3.8 The coordinate system of planed visible side

When the visible side of the target object consists of more than one surface (different planes) as shown in Fig. 3.9, every surface (plane) will have its coordinate system. The main coordinate system will belong to the surface where the contact or the grasping task will be performed.

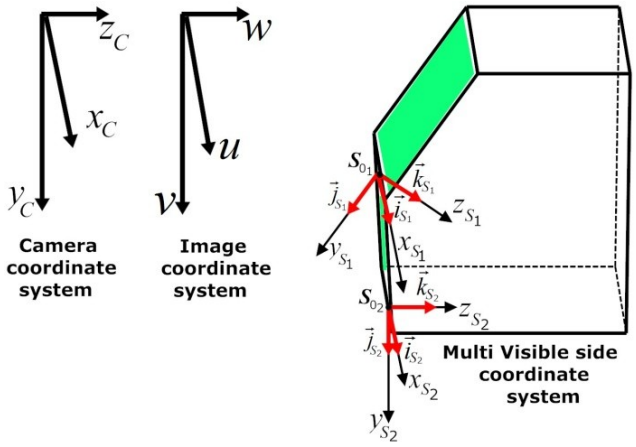

Fig. 3.9 Visible side consists of multi surfaces

2. *Visible side as curved surface*

In this case the visible side has a curved surface. Fig. 3.10 presents three objects which have different shapes of visible sides. Let's define the visible side coordinate system in a point P of the target object as follows: The axis y_S will be the tangent in the direction of the major axis of

the visible side, the axis x_S will be tangent which is in the direction of the minor axis of the visible side of the object and the axis z_S will be the perpendicular axis (normal) to the plane (x_S, y_S) according to the right-hand rule. The strategies of choosing the point P and the algorithms of calculation of the tangent and normal will be discussed later.

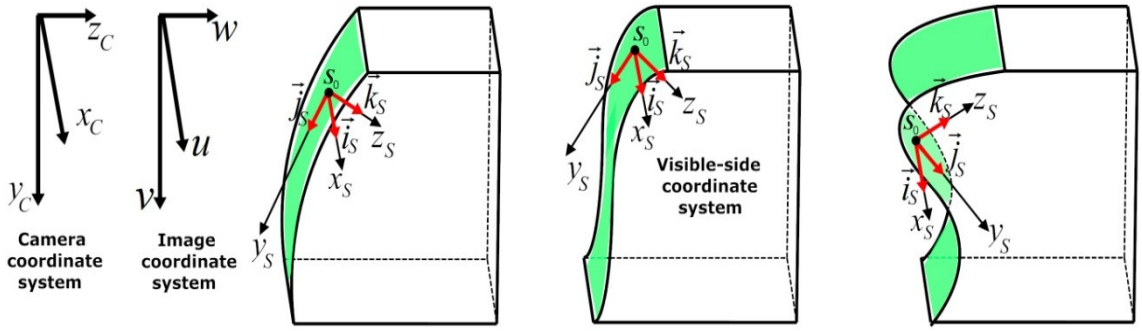

Fig. 3.10 Curved visible side

3.1.3 Transformation between coordinate systems

This section will represent the poses of the robot's tool and the target object relative to the camera coordinate system and show how the transformation between them will be performed.

3.1.3.1 Transformation between camera and tool coordinate systems

As shown previously in Fig. 3.6, the Kinect camera is camera-in-hand meaning that the relation between camera coordinate system and tool coordinate system is constant and it consists only of translation part (there is no rotation difference between both coordinate systems).

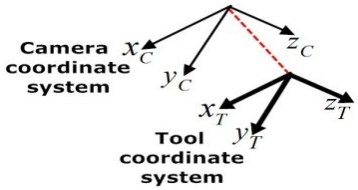

Fig. 3.11 Camera/Tool translation

If we assume that $P_C = (x_C, y_C, z_C)$ and $P_T = (x_T, y_T, z_T)$ are the coordinate of point P relative to camera and tool coordinate system, the transformation from one frame to other can be performed as follow:

$$P_T = \begin{bmatrix} 1 & 0 & 0 & {}^{T}x_C \\ 0 & 1 & 0 & {}^{T}y_C \\ 0 & 0 & 1 & {}^{T}z_C \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot P_C \qquad (3.6)$$

3.1.3.2 Transformation between camera and image coordinate systems

The 3D Kinect camera and 2D Sony camera which are used in this work have pinhole-model. Pinhole camera is a simple lensless camera provided with a tiny hole on one side and a film or

photographic paper on the other. The light passes through the hole and an image will be formed in the camera.

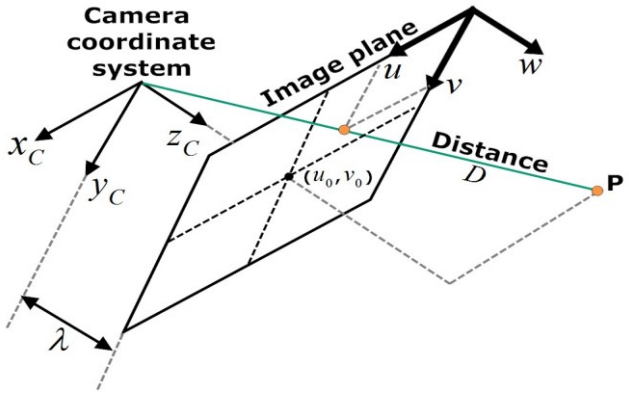

Fig. 3.12 Central-perspective imaging model

Fig. 3.12 illustrates the relative position of one point of the target object according to the camera coordinate system (x_C, y_C, z_C). A non-inverted image is projected onto the image plane (u, v) which is located at $z_c = \lambda$. The point P in the world coordinate system is projected to the image plane. The projection of a scene on an image plane will lead to the loss of the depth information of the scene. In other words, the definition of projection is a method of mapping the points of 3D space to a 2D plane. The projection method used in pinhole cameras is called perspective projection. Perspective means that the objects will appear smaller whenever it is farther from the camera. Using similarity of triangles it can be written:

$$\frac{u}{\lambda} = \frac{x_C}{z_C} \tag{3.7}$$

$$\frac{v}{\lambda} = \frac{y_C}{z_C} \tag{3.8}$$

From (3.7) and (3.8) it can be written:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{\lambda}{z_C} \begin{bmatrix} x_C \\ y_C \end{bmatrix} \tag{3.9}$$

Equation (3.9) could be written in different way:

$$\begin{bmatrix} u \cdot z_C \\ v \cdot z_C \end{bmatrix} = \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \end{bmatrix} \begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix} \tag{3.10}$$

By rewriting (3.10) in homogenous form:

$$\begin{bmatrix} u \cdot z_C \\ v \cdot z_C \\ z_C \end{bmatrix} = \begin{bmatrix} \lambda & 0 & 0 & 0 \\ 0 & \lambda & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_C \\ y_C \\ z_C \\ 1 \end{bmatrix} \tag{3.11}$$

As shown in Fig. 3.12 the coordinate system of the image plane is not in the central point (u_0, v_0) (where the optical axis are intersected with the sensor plane) of the plane, but it is in

the upper left corner of the image plane. In addition to that, if the x_C-axis focal length is not equal to the focal length in y_C-axis, (3.11) will be written as follows:

$$\begin{bmatrix} u \cdot z_C \\ v \cdot z_C \\ z_C \end{bmatrix} = \begin{bmatrix} \lambda_x & 0 & u_0 & 0 \\ 0 & \lambda_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_C \\ y_C \\ z_C \\ 1 \end{bmatrix} \tag{3.12}$$

Another factor could be added to (3.12) when the image coordinate axes are skew this factor will be called S_C. By assuming $S = z_C$, (3.12) could be written as follows:

$$S \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \lambda_x & S_C & u_0 & 0 \\ 0 & \lambda_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_C \\ y_C \\ z_C \\ 1 \end{bmatrix} \tag{3.13}$$

Transformation from camera coordinate system to the image plane coordinate will be performed using intrinsic parameter matrix as shown in (3.13).

$$K = \begin{bmatrix} \lambda_x & S_C & u_0 \\ 0 & \lambda_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$

The transformation from world coordinate system to the camera coordinate system will be performed by translation and rotation matrix as follows:

$$\begin{bmatrix} x_C \\ y_C \\ z_C \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_W \\ y_W \\ z_W \\ 1 \end{bmatrix} \tag{3.15}$$

Hence, the transformation from world coordinate to the image plane will be as follows:

$$S \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \lambda_x & S_C & u_0 & 0 \\ 0 & \lambda_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_W \\ y_W \\ z_W \\ 1 \end{bmatrix} \tag{3.16}$$

Many papers and reports have focused on finding the performance curves and intrinsic parameters of the Kinect camera such as (Freedman, Shpunt, Machline, & Arieli, 2010) and (Viager, 2011). In the last one the intrinsic parameters of Kinect camera are given as follow:

$$K_{IR} = \begin{bmatrix} 585.6 & 0 & 316 \\ 0 & 585.6 & 247.6 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.17}$$

$$K_{RGB} = \begin{bmatrix} 524 & 0 & 316 \\ 0 & 524 & 238.5 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.18}$$

Equation (3.17) presents the intrinsic parameters of the IR camera in the Kinect sensor, while (3.18) presents the intrinsic parameters of the RGB camera.

3.1.3.3 Transformation between visible side and image coordinate systems

Fig. 3.13 shows some examples with different poses of the visible side planes according to the camera coordinate system. In the first case the visible side (S) coordinate system is parallel to the camera (C) coordinate system. In the second case there is the rotation about x_S, whereas in the third case the rotation is about y_S direction.

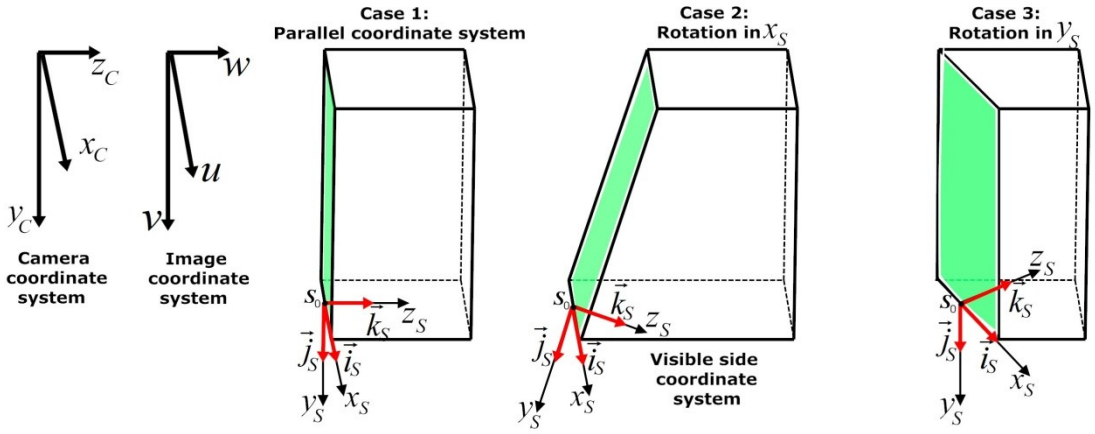

Fig. 3.13 Examples of visible side coordinate system

The transformation from S coordinate system to the image plane coordinate system will be performed depending on perspective projection matrix as follows:

In the first case of Fig. 3.13, where the S coordinate system is parallel to the camera coordinate system:

$$S \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \lambda_x & S_C & u_0 & 0 \\ 0 & \lambda_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_S \\ y_S \\ 0 \\ 1 \end{bmatrix} \tag{3.19}$$

In the second case of Fig. 3.13, where the rotation is about x_S direction:

$$S \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \lambda_x & S_C & u_0 & 0 \\ 0 & \lambda_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & cos\theta_x & -sin\theta_x & t_y \\ 0 & sin\theta_x & cos\theta_x & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_S \\ y_S \\ 0 \\ 1 \end{bmatrix} \tag{3.20}$$

In the third case of Fig. 3.13, where rotation is about y_S direction:

$$S \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \lambda_x & S_C & u_0 & 0 \\ 0 & \lambda_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} cos\theta_x & 0 & sin\theta_x & t_x \\ 0 & 1 & 0 & t_y \\ -sin\theta_x & 0 & cos\theta_x & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_S \\ y_S \\ 0 \\ 1 \end{bmatrix} \tag{3.21}$$

3.2 Concept of 4X2D visual servoing approach

This section will handle the main concept of suggested approach and the visual servoing.

3.2.1 Relative position between visible side and camera coordinate systems

Fig. 3.14 shows the relative position of the origin of the visible side frame S_0 with respect to the camera coordinate system. Kinect camera will deliver the projected position of S_0 into the image plane (u, v) and the distance D between both coordinate systems.

As previously, it can be written:

$$u = \lambda \cdot \frac{x_c}{z_c} \tag{3.22}$$

$$v = \lambda \cdot \frac{y_c}{z_c} \tag{3.23}$$

where (x_c, y_c, z_c) coordinate of point S_0 relative to camera coordinate system, u and v can be calculated from the image plane and λ is camera focal length.

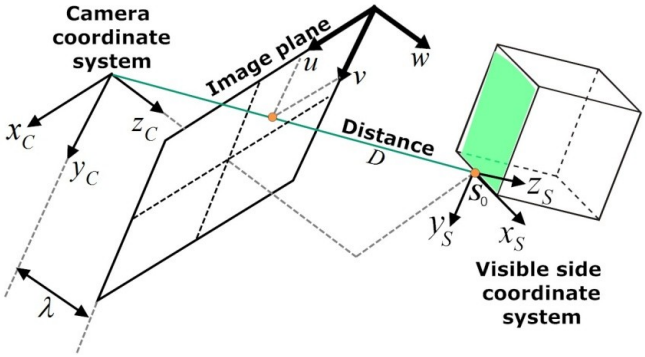

Fig. 3.14 Visible side and camera coordinate system

From the depth image the distance D between the origin S_0 and the camera coordinate system, it can be written:

$$D^2 = x_c^2 + y_c^2 + z_c^2 \tag{3.24}$$

In equations (3.22) (3.23) and (3.24) four values are known u, v, D, λ and three others are unknown x_c, y_c and z_c. If we could calculate these three values, we can define the relative position between the visible side and the camera coordinate systems. The previous equations could be rewritten as follow:

$$x_c = z_c \cdot \frac{u}{\lambda} \tag{3.25}$$

$$y_c = z_c \cdot \frac{v}{\lambda} \tag{3.26}$$

$$z_c^2 = D^2 - x_c^2 - y_c^2 \tag{3.27}$$

By substituting (3.25) and (3.26) in (3.27):

$$z_c^2 = D^2 - (z_c \cdot \frac{u}{\lambda})^2 - (z_c \cdot \frac{v}{\lambda})^2 \tag{3.28}$$

$$z_c^2 + \frac{u^2}{\lambda^2} \cdot z_c^2 + \frac{v^2}{\lambda^2} \cdot z_c^2 = D^2$$

$$z_c^2 = \frac{D^2}{1 + \frac{u^2}{\lambda^2} + \frac{v^2}{\lambda^2}}$$

$$z_c = \pm \frac{\lambda \cdot D}{\sqrt{\lambda^2 + u^2 + v^2}} \tag{3.29}$$

By substituting (3.29) in (3.25), x_c can be calculated:

$$x_c = \frac{\lambda \cdot D}{\sqrt{\lambda^2 + u^2 + v^2}} \cdot \frac{u}{\lambda} = \frac{D \cdot u}{\sqrt{\lambda^2 + u^2 + v^2}} \tag{3.30}$$

By compensation (3.29) in (3.26), y_c can be calculated:

$$y_c = \frac{\lambda \cdot D}{\sqrt{\lambda^2 + u^2 + v^2}} \cdot \frac{v}{\lambda} = \frac{D \cdot v}{\sqrt{\lambda^2 + u^2 + v^2}} \tag{3.31}$$

From (3.29) (3.30) and (3.31), the relative position of the origin S_0 with respect to the camera coordinate system (x_c, y_c, z_c) can be calculated.

3.2.2 Relative orientation between visible side and camera coordinate systems

This section presents how to calculate the orientation of the object with respect to the camera coordinate system. According to the approach suggested in this work, the orientation of the object will be calculated from three 2D images (u, v), (u, w), and (w, v) with the help of image moments. First of all we will explain how to calculate the 2D orientation of an object with the help of image moments after that we will generalize this concept to be used in the three images (3D spaces of images).

3.2.2.1 Object's orientation in image plane

The orientation of the object in 2D image plane is defined as the angle between the horizontal axis and the axis, where the object can be rotated with minimal inertia (i.e. the direction of the major semi-axis). In this direction the object has its longest extension. Fig. 3.15 presents the projection of an object in the (u, v) image plane.

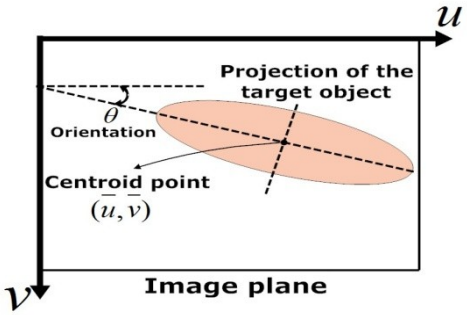

Fig. 3.15 Orientation of the target object

Let's assume that the target object found in the current image $I(u,v)$ which is shown in the Fig. 3.15 is represented by the indicate function $S(u,v)$. $S(u,v)=1$, when pixel belongs to the target object, otherwise $S(u,v)=0$. There are two types of moments: non-central moments m_{pq} and central moments μ_{pq}:

$$m_{pq}=\sum_{v=0}^{V-1}\sum_{u=0}^{U-1}u^p\cdot\ v^q\cdot S(u,v) \tag{3.32}$$

where the order of the moment is $(p+q)$, so when $p=0$ and $q=0$ the m_{00} moment can be calculated as follows:

$$m_{00}=\sum_{v=0}^{V-1}\sum_{u=0}^{U-1}u^0\cdot\ v^0\cdot S(u,v)=\sum_{v=0}^{V-1}\sum_{u=0}^{U-1}S(u,v) \tag{3.33}$$

where m_{00} is the total number of pixels belonging to the object. In the same way the centroid point of the object can be calculated:

$$\bar{u}=\frac{m_{10}}{m_{00}} \tag{3.34}$$

$$\bar{v}=\frac{m_{01}}{m_{00}} \tag{3.35}$$

The central moments of the object $S(u,v)$ can be calculated:

$$\mu_{pq}=\sum_{v=0}^{V-1}\sum_{u=0}^{U-1}(u-\bar{u})^p\cdot\ (v-\bar{v})^q\cdot S(u,v) \tag{3.36}$$

The main advantage of central moments is that they are invariant to translations of the object. The disadvantage of the central moments is their dependency on the size of the object (scale). Usually, the area of the object could be used as a scaling factor. Hence, by dividing the central moments with power of the area of the object, we get the central normalized moments ν_{pq}:

$$\nu_{pq}=\frac{\mu_{pq}}{m_{00}^{(\frac{p+q}{2}+1)}} \tag{3.37}$$

The main advantage of this kind of moments is their invariancy to the scale and translations. Using the central moments, the orientation of the 2D object can be calculated as follows (Suchý, 2013):

$$\theta=\frac{1}{2}arctan\left(\frac{2\cdot\mu_{11}}{\mu_{20}-\mu_{02}}\right) \tag{3.38}$$

3.2.2.2 Object's orientation in the 3D image space

As shown previously, in our experiment we have supposed that the camera coordinate system is parallel to the tool coordinate system, which means the object's visible side seen by the camera will take the main role to specify the strategies of performing the grasping or

contacting tasks. Therefore, this side will be projected on the two other planes (u, w) and (w, v). The projections of the visible side to the two other planes will be calculated by combining the RGB and depth images. These projections will give us the opportunity to achieve the contacting or grasping tasks with easier way to extract the vision error between the actual and target position of the object. In the next section, three different examples will be presented to get better overview of visible side's projection.

Example 1: Rotation about z_S

In this example, we will suppose that the transformation between visible side and image coordinate systems contains rotation about z_S only, as shown in Fig. 3.16.

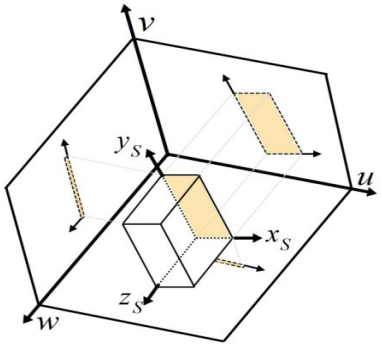

Fig. 3.16 Represented object in 3D image space (Example 1: Rotation about z_S)

Rotation about only z_S direction means that the visible side plane (orange plane) stays parallel to (u, v) plane and all points of the visible side have almost the same depth distance. The projection of the visible side on the two other planes (u, w) and (w, v) will be performed by combining the RGB and depth images. Hence, (u, w) image plane will represent the relation between x_S direction and the depth z_S, whereas (w, v) image plane will represent the relation between y_S direction and the depth z_S.

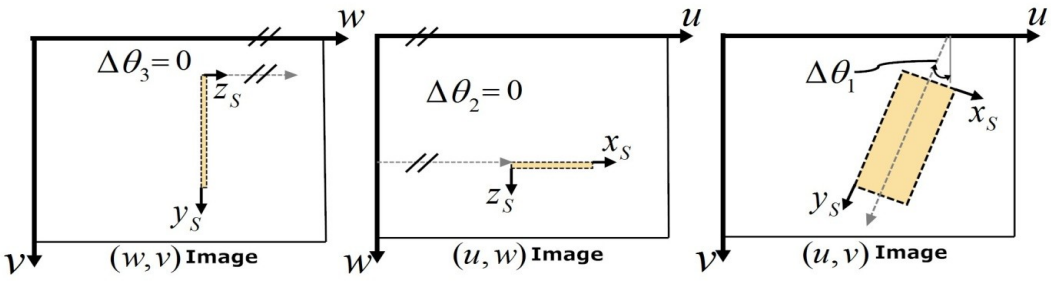

Fig. 3.17 Projections of the visible side (Example 1: Rotation about z_S)

As shown in Fig. 3.17 there are three 2D objects which are the projections of the visible side of the object in the 3D image space. Hence, there are three angles $\Delta\theta_1$, $\Delta\theta_2$ and $\Delta\theta_3$ that can be calculated from three image planes. $\Delta\theta_1$ is the rotation angle between the projection of visible side coordinate system (x_S, y_S) and the image plane (u, v). $\Delta\theta_2$ is the rotation angle between the projection of visible side coordinate system (x_S, z_S) and the image plane (u, w). $\Delta\theta_3$ is the rotation angle between the projection of visible side coordinate system (z_S, y_S) and the image

plane (w, v). As discussed previously, using the central moments of the image (see (3.38)) and depending on the definition of the 2D object's orientation (the angle between the horizontal axes and the direction of the major semi-axis), the orientation of the 2D objects in three image planes will be as follows:

- The orientation of the 2D projection of the object in the (w, v) image plane is $\theta_3 = 90^{\circ}$(angle between major semi-axis of the object y_S and the horizontal axis w). However, the rotation angle should be measured between y_S and v axes or z_S and w axes. Hence, the rotation angle $\Delta\theta_3 = 90^{\circ} - 90^{\circ} = 0$.
- The orientation of the 2D projection of the object in the (u, w) image plane is $\theta_2 = 0$ (major axis x_S is parallel to horizontal axis u). Hence, the rotation angle $\Delta\theta_2 = 0$.
- The orientation of the 2D projection of the object in the (u, v) image plane is $\theta_1$which can be calculated by (3.38):

$$\theta_1 = \frac{1}{2} arctan\left(\frac{2 \cdot \mu_{11}}{\mu_{20} - \mu_{02}}\right).$$

Hence, the rotation angle which makes the (x_S, y_S) plane parallel to the (u, v) image plane is $\Delta\theta_1 = \theta_1$

Example 2: Rotation about x_S

In this example, we will suppose that the transformation between visible side and image coordinate systems contains rotation about the direction x_S only, as shown in Fig. 3.18. As a result of rotation about x_S, the visible side which is projected onto (u, v) image plane consists of two planes. Hence, there are two coordinate systems of the visible side. Depending on the target task and where the contact point between the object and the robot will be arisen, the robot system will choose the fit S coordinate system.

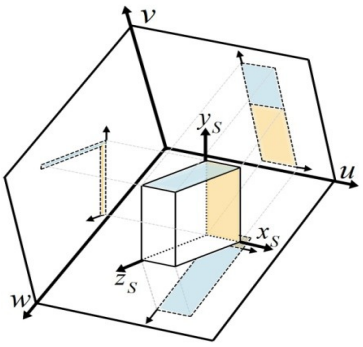

Fig. 3.18 Projections of the visible side (example 2: rotation about x_S)

In the same way, the rotation angles between projections planes and image planes coordinate systems can be calculated:

- $\Delta\theta_3 = \theta_3$, the rotation angle which makes the (z_S, y_S) plane parallel to the (w, v) image plane.
- $\Delta\theta_2 = 0$, the (x_S, z_S) plane and the (u, w) image plane are parallel.
- $\Delta\theta_1 = 0$, the (x_S, y_S) plane and the (u, v) image plane are parallel.

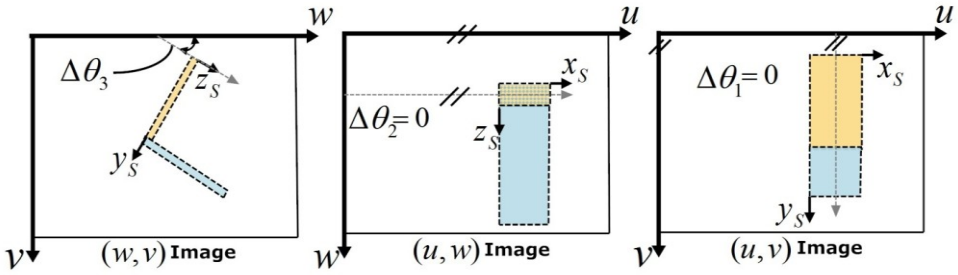

Fig. 3.19 Projections of the visible side (Example 2: Rotation about x_S)

Example 3: Rotation about y_S

In this example, we will suppose that the transformation between visible side and image coordinate systems contains rotation about the direction y_S only, as shown in Fig. 3.20. As a result of rotation about y_S, the visible side which is projected onto (u, v) image plane consists of two planes. Hence, there are two coordinate systems of the visible side. In the same way, depending on the proposed task and where the robot should contact the objet, the robot system can choose the visible side coordinate system.

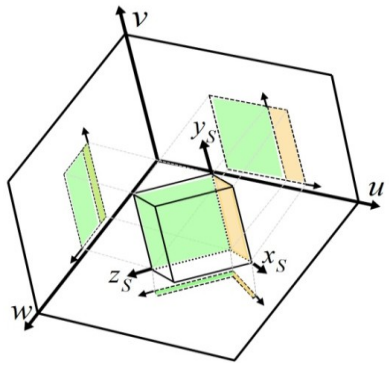

Fig. 3.20 Projections of the visible side (Example 3: Rotation about y_S)

As shown previously, the rotation angles between projections planes and image planes coordinate systems can be calculated:

- $\Delta\theta_3 = 0$, the (z_S, y_S) plane and (w, v) image plane are parallel, $\Delta\theta_3 = \theta_3$.
- $\Delta\theta_2 = \theta_2$, the rotation angle which makes the (x_S, z_S) plane parallel to the (u, w) image plane.
- $\Delta\theta_1 = 0$, the (x_S, y_S) plane and the (u, v) image plane are parallel.

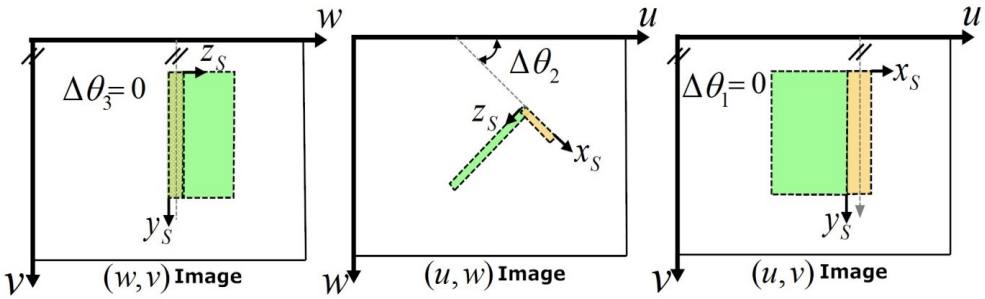

Fig. 3.21 Projections of the visible side (Example 3: Rotation about y_S)

3.3 Control error of 4X2D visual servoing approach

The first step of the visual servoing is defining in any image a particular set of features that describes the goal which should be reached. In the image-based visual servoing approach, the determination of the visual features is still considered as open problem. In other words, how does the system choose the most appropriated visual features in the control scheme in order to obtain an optimal behavior of the system? Previously, many papers have used the coordinates of points or parameters which describe the segments of the image, e.g. straight line, ellipses (Espiau, Chaumette, & Rives, 1992), (Hutchinson, Hager, & Corke, 1996). In other works (Chaumette, 2004) and (Chaumette, 2002), it has been attempted to use moments in image-based visual servoing*.

Actually, the image moments have been widely used in pattern recognition and visual servoing for a long time (Prokop & Reeves, 1992). However, the main problem of using image moments in visual servoing loop was that the analytical form of the interaction matrix related to the image moments was unavailable. In (Bien, Jang, & Park, 1993) the image moments implemented to control only four degrees of freedom of a robot (x, y, z, θ_z) were: the area, the centroid and the main orientation of an object in the image. These four visual features (coordinates x_g and y_g of the center of gravity, area a and orientation Θ) can be easily obtained from moments of order less than 3 which are closely related to the velocities of the camera v_x and w_y, v_y and w_x, v_z, and w_z respectively. However, using the previous image moments leads to the situation that the robot will execute trajectories which are desirable in the image and which can be indirectly and seemingly contorted in Cartesian space. This potential problem appears when redundant image points coordinates are used, e.g. coupled features.

The work (Corke & Hutchinson, 2001) has overcome this problem by decoupling the z-axis rotational and translational components of the control from the remaining degrees of freedom, as follows:

$$\dot{\vec{f}} = J_{xy}\dot{\vec{r}}_{xy} + J_z\dot{\vec{r}}_z \tag{3.39}$$

where $\dot{\vec{r}}_{xy} = [v_x\ v_y\ w_x\ w_y]$, $\dot{\vec{r}}_z = [v_z\ w_z]$, J_{xy}, J_z are respectively components or columns 1,2,4,5 and 3,6 of Jacobian matrix J and $\dot{\vec{f}}$ is the feature point coordinate error. Hence,

$$\dot{\vec{r}}_{xy} = J_{xy}^{+}\{\dot{\vec{f}} - J_z\dot{\vec{r}}_z\} \tag{3.40}$$

The Z-axis velocity will be based directly on two features (orientation and area of the object).

$$v_z = \gamma_{v_z}(a^* - a),\ \ w_z = \gamma_{w_z}(\theta_z^* - \theta_z) \tag{3.41}$$

* The calculation of the general interaction matrix of image moments is illustrated in Appendix I. Furthermore, in Appendix I the interaction matrix of coordinates of the center of gravity (x_g, y_g), the area (a) and the orientation (Θ) of the target object will be calculated.

where γ_{v_z}, γ_{w_z} are the scalar gain coefficient. Another approach (Chaumette, 2002) has used two supplementary visual features in order to decouple w_y from v_x and w_x from v_y. These features are based on classical skewness moments $s_x = \mu_{30}/\mu_{20}^{3/2}$ and $s_y = \mu_{03}/\mu_{03}^{3/2}$. For the symmetrical objects, they have proposed:

$$\begin{cases} s_x = \sqrt{a}(s_1 t_1 + s_2 t_2)/K \\ s_y = \sqrt{a}(s_2 t_1 - s_1 t_2)/K \end{cases} \tag{3.42}$$

$$where \begin{cases} s_1 = \mu_{03} - 3\mu_{21}, \quad t_1 = (\mu_{20} - \mu_{02})^2 - 4\mu_{11}^2 \\ s_2 = \mu_{30} - 3\mu_{12}, \quad t_2 = 4\mu_{11}(\mu_{20} - \mu_{02}) \\ K = \Delta(\mu_{20} + \mu_{02})^{3/2} \end{cases} \tag{3.43}$$

For non symmetrical objects:

$$\begin{cases} p_x = \Delta/(\mu_{20} + \mu_{02})^2 \\ p_y = a(s_1^2 + s_2^2)/(\mu_{20} + \mu_{02})^3 \end{cases} \tag{3.44}$$

Hence, the vector of visual features used for tracking symmetrical objects is as follows:

$$s = (x_g, y_g, a, s_x, s_y, \theta_z) \tag{3.45}$$

whereas the vector of visual features used for tracking the nonsymmetrical objects is:

$$s = (x_g, y_g, a, p_x, p_y, \theta_z) \tag{3.46}$$

For the proposed features and for the configurations where the object is parallel to the image plane at the desired position, the interaction matrix has shown nice decoupling feature and the results are satisfactory even if the object is not parallel to the image plane at the beginning of the positioning task. In this approach, the maximum order of the implemented moments is the third order. However, whenever the used moments have less order then the system will be more robust and it will have more numerical stability. Because of that, one advantage of the proposed approach in this chapter that the control loop can use the calculated orientations from (u, w) and (w, v) images instead of s_x and s_y which can be easily calculated from the second order moments.

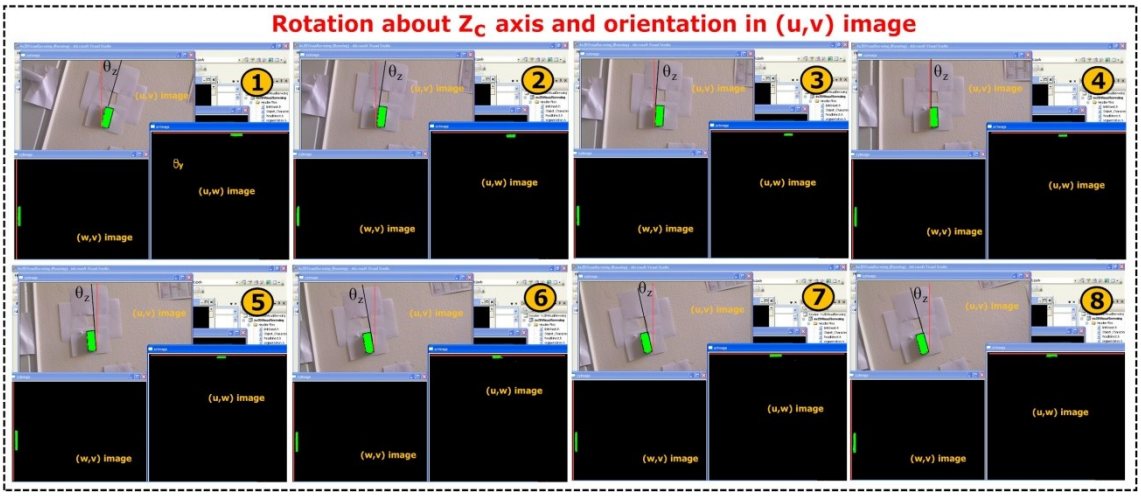

Fig. 3.22 Relation between rotation about Z_c and orientation in (u,v) image

Fig. 3.22, Fig. 3.23 and Fig. 3.24 illustrate some experimental results during the rotation of the camera about different axes. The target object and its projections are of green color. Fig. 3.22 shows the proportional changing of the orientation in (u, v) image during the rotation of the camera about axis $\mathrm{Z_c}$. As shown, there is nice decoupling between the rotation about $\mathrm{Z_c}$ and the orientation in (w, v) and (u, w) images, they have constant values.

In the same way, the experiment will be repeated during the rotation of the camera about axis $\mathrm{Y_c}$, as shown in Fig. 3.23. Only the orientation in (u, w) image will be changed, whereas orientations in (u, v) and (w, v) images are constant and they are decoupled from the rotating about $\mathrm{Y_c}$.

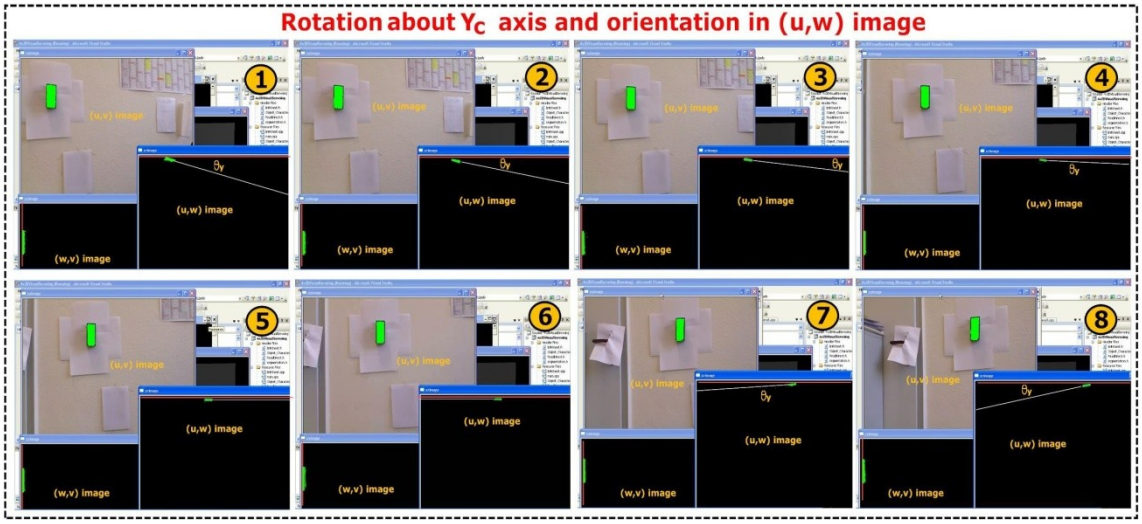

Fig. 3.23 Relation between rotation about Y_c and orientation in (u,w) image

In Fig. 3.24, the experiment will be performed during the rotation of the camera in $\mathrm{X_c}$ and only the orientation in (w, v) image will be changed, whereas the orientations in (u, v) and (u, w) images are constant and they are decoupled from the rotating in $\mathrm{X_c}$.

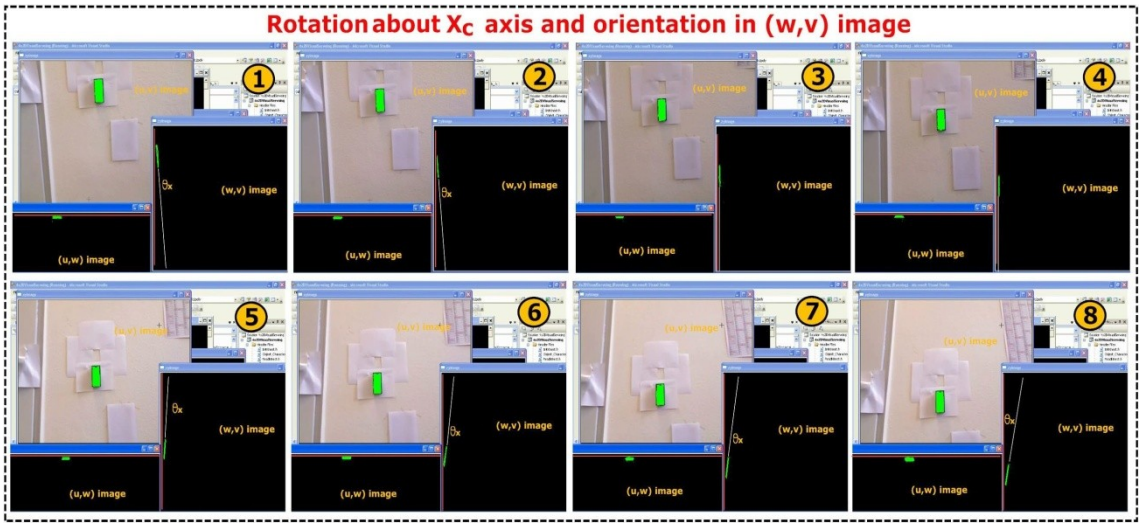

Fig. 3.24 Relation between rotation about X_c and orientation in (w,v) image

Fig. 3.25 presents the projections of the target object in different poses on the four images: (depth, (u,v), (u,w) and (w,v)). The target object is green, whereas the blue part indicates that this side of the object has higher position (object is diagonally positioned). This part of the target object is the position where the robot will grasp the object, because it is the nearest part of the object from the camera (robot tool). It is clear that one can directly calculate the inclination angle of the object either in the first case from (w,v) image or in the second case from (u,w) image.

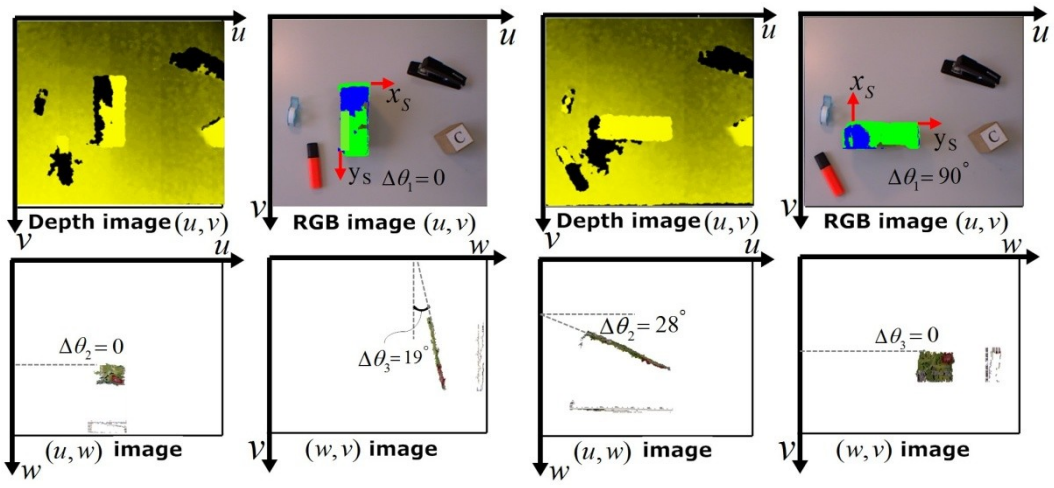

Fig. 3.25 Different poses of the target object

To conclude, this chapter has illustrated some previous work of the visual servoing and it has proposed a new visual servoing approach which is suitable for the new generation of the RGBD camera (Kinect camera). 4x2D visual servoing has combined the correspondence of color and depth images to build two new images. Depending on measured orientations in (u,v), (u,w) and (w,v) images, the control error signals will be calculated in order to track any flat surface. More results of visual servoing will be illustrated in Chapter 5.

Chapter 4

Automatic Decision System

In recent years there have been rapidly increased development of robotic systems and their applications. The challenge today is e.g. that the robot performs different successive subtasks to achieve one or more complicated tasks similar to human. In advanced robotic application using only one kind of feedback is sometimes insufficient to achieve the desired goals perfectly. In order to get more information about the work environment it is preferable to use different kinds of sensors such as vision sensor, force sensor, acceleration sensor, tactile sensor etc. However, from the point of view of control, more sensors mean more possibilities for the structure of the control system. In fact, it can be found in scientific papers a number of control algorithms and different structures for the robot control and by using more sensors more approaches will be appeared, an illustration of some approaches to position/force control are impedance control, which uses relationships between acting forces and robot poses to adjust the mechanical impedance of the end-effector to external forces; parallel control (the parallel approach to force/position control of robotic robots), which enables to control both force and pose, along the same task space direction; hybrid position/force control, which controls force and pose in two orthogonal directions or even in more different directions.

As is known, vision and force sensors are the most common external sensors in robotic applications, because of that this chapter will introduce an automatic decision system which decides automatically the most appropriated vision/force control structure for different tasks depending on the image information, a priori-knowledge and stop-conditions of the tasks. Furthermore, it will redefine the vision/force control modes in a way that makes the robot system able to define the most appropriated mode for every direction automatically. This chapter will use all possible types of vision/force control combinations and it also use different control structures in different directions (linear/ rotational directions of the visible side frame (see Chapter 3)) in one task to insure better performance of the robot during the task. Theoretically, the automatic decision system can decide automatically the type of the vision/force control structure, if it can answer on the following questions:

- How many directions in robot should be controlled?
- Which directions should be vision, commanded position or force controlled?

- How to insure that the feedback information could be used reliably?
- How to define what is the most appropriate vision/force control mode?

These questions will be called the pivotal questions. If the proposed automatic decision system can answer these pivotal questions, it will help the robot to:

- Perform different successive and complex tasks
- Grasp/contact imprecisely objects or environments which have different poses
- Decide automatically the most appropriated combination of vision/force feedback for every task
- Benefit from all the advantages of different vision/force control structures.
- React immediately on the changes from one control cycle to another one because of occurrence of some unforeseen events.
- Reduce the human intervention or reprogramming during the execution of the task.

This chapter will introduce an automatic decision system which can answer on the previous pivotal question based on analyzing and combining of: 1. Basic information about sensor, task and object provided by user. 2. Scene properties extracted by vision system.

4.1 Related work

The most relevant works related to the proposed system in this chapter are the work (Finkemeyer, Kröger, & Wahl, 2010) and the work (Baeten, Bruynicks, & Schutter, 2003), where the authors have performed an intensive research in the field of sensor-guided motion and sensor integration. In (Finkemeyer, Kröger, & Wahl, 2005), they have proposed manipulation of primitive nets as output of task planning systems. These tasks will be decomposed into single manipulation primitive used to generate the parameters for the hybrid control. Furthermore, they have introduced software architecture to realize the complex control structure for compliant motion. The implementation of the control architecture is based on the middleware MiRPA (Middleware for Robotic and Process Control Applications). In other work (Kröger & Wahl, 2008), they have concentrated on sensor integration and online trajectory generation for improving the reaction of the robot to unforeseen events using hybrid switched control. However, the common hybrid switched controllers facing the problem of switching between different controllers in the real time, this could lead to the following question: How can the control system stay stable during the switching? How can the robot be controlled, if its measurements are out the range of the currently used sensor? For example, force control loop will be open in the free space, in the same way the visual servoing loop will be also open when the target object is out of the camera view. To solve all these problems in the hybrid switched systems, they have introduced the adaptive selection matrix (Finkemeyer, Kröger, & Wahl, 2010) and (Kröger & Finkemeyer, 2011). The adaptive selection matrix maps the parameters of the manipulation primitive to low-level control layers including the hybrid switched-system control structure.

As shown in Fig. 4.1, the proposed selection matrix is not static during one robot command: It considers the current robot and environment state. Thus it is an adaptive selection matrix. In general, in hybrid force/position system the classical selection matrix (2D) is implemented. However, for the complicated task where different types of sensors will be implemented and the control loop will be switched frequently, the simple two-dimensional classical selection matrix will be insufficient.

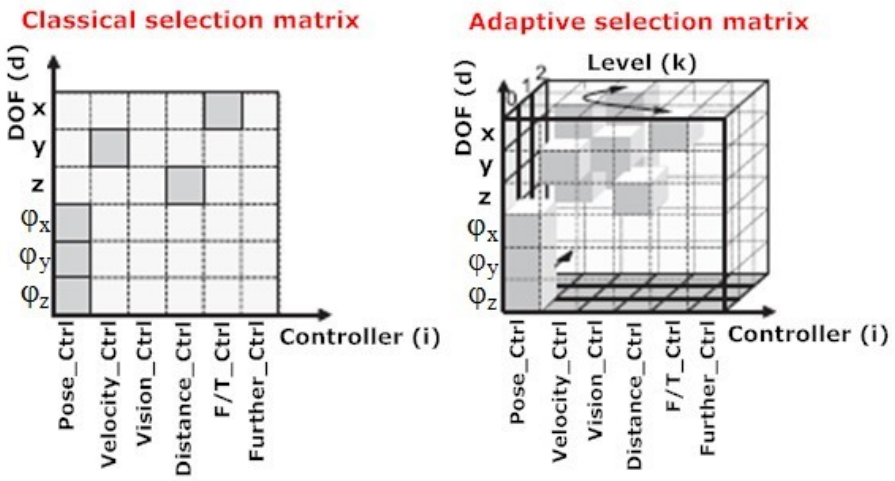

Fig. 4.1 Classical and adaptive selection matrix (Finkemeyer, Kröger, & Wahl, 2010)

The implementation of different and alternative control loops requires the extension which is shown in the third dimension (control level). In this case, the selection matrix will work dynamically and depend on the currently available sensors/controllers and the assignment of controllers. However, there are some points still open for discussion:

- How the inexperienced user can easily describe the task or the priori-knowledge about the task and the object in such somewhat more complicated system? As is known, whenever the system is more complicated, more wrong information could be provided by the user.
- The work (Finkemeyer, Kröger, & Wahl, 2010) tries to generalize the adaptive selection matrix in order fit several kinds of controllers and sensors. However, in vision and force sensor fusion, there are different control structures such as traded, shared, hybrid, etc. How the adaptive selection matrix can define the most appropriated vision/force control mode for the proposed task?
- Furthermore, in the same work, the controller of the task will depend on the actual robot state variables and the available sensor signals. However, how the robot system can automatically define if the sensor signals are available or not? Furthermore, if they are available, can they reliably be used? In other words, how the robot can benefit from all the available information provided by the sensors in useful way?
- How the system will define the switching conditions automatically and the time of switching from one control mode to the other?

This chapter will handle all the previous points as follows:

- It will propose a user interface which consists of simple questions called basic information which any user can answer before starting the task. The user interface will make the description easier for the user. Moreover, it assumes a new coordinate system (approach) which is convenient to human conception.
- It will concentrate only on the fusion of vision and force control, and it will illustrate how the robot can automatically define the most appropriate vision/force control structure.
- It will illustrate how the robot system can use all the available information which could be provided by the vision sensor, not only for the target object but also for the whole scene. In other words, this work will not use the vision system as simple feedback or as a desired position estimator but it will use it to extract the properties of the scene and to understand the surrounding circumstances of the object during performing the task.
- It will propose automatic decision system which will combine all the previous information and it will control the values of the selection matrix. Hence, the contributions of this chapter could be in the future combined with the work (Finkemeyer, Kröger, & Wahl, 2010) to form an automatic adaptive selection matrix.

Next section will illustrate the automatic decision system in details.

4.2 Description of the automatic decision system

Fig. 4.2 presents the overview scheme of the automatic decision system which consists of three main interfaces: User interface, middle interface and robot interface. With the help of user interface, the user can provide the control system with the simple basic information. This information will be called basic information. The middle interface will analyze this information and then it will combine the basic information with the extracted properties of the scene by the sensor, i.e. vision information, in order to answer the pivotal questions (illustrated later) and decide automatically the vision/force control structure. After that, the middle interface will control the values of the selection matrixes as output signals with the help of the automatic decision module. The values of selection matrices in robot interface will define which directions will be vision controlled and which will be force controlled. As shown in Fig. 4.2, the user interface consists of two components of: the priori-knowledge and the stop conditions. The middle interface consists of two parts: 1. Information combining and analyzing module. 2. Automatic decision module. The robot interface contains three selection matrixes for defining the vision/force control structure.

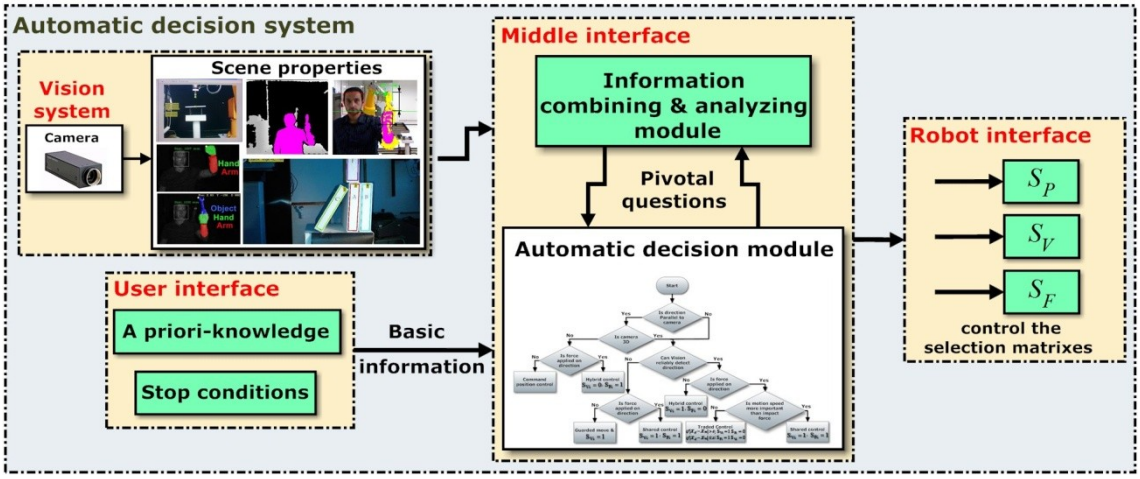

Fig. 4.2 Overview of the automatic decision system

Notation

The following notations are used: With $\vec{X}^T = [x, y, z, \Theta_x, \Theta_y, \Theta_z]$ will be further denoted the pose vector of end-effector; with small letters m, d and v the sources of information about pose vector $\vec{X}$, e.g. measured $\vec{X}_m$, desired $\vec{X}_d$ or vision $\vec{X}_v$ are denoted and with capital letters C, T and S the coordinate system for vector $\vec{X}$: camera, task and visible side frame (the visible side of the object by the camera, see chapter 3) are denoted, e.g. x_{mC} denotes the measured position of x axis relative to the camera coordinate system. S_P, S_V and S_F are position, vision and force selection matrixes. The automatic decision algorithm will define the structure for fusing vision and force control in all directions x, y, z and orientations Θ_x, Θ_y and Θ_z. It has to be considered, that the control values which are measured in the sensor frame e.g. in vision or force sensor frame, are transformable to the visible side frame. In the next sections we will

illustrate in details all components of the automatic decision system starting from the user interface and ending with robot interface.

4.3 User interface

Human intervention in robots during the task execution cannot be neglected, especially when the robot performs different complex or successive tasks. However, in many cases making the decision about the robot control structure and changing execution algorithms is not always clear for the inexperienced user. Our idea is to reduce the human intervention during the task execution as much as possible by defining some simple questions which any user can answer before starting the task. These questions will be called the basic questions. The automatic decision module will analyze these answers and it will combine them with other information in order to answer the pivotal questions and decide automatically during the execution of complex and successive tasks. As shown in Fig. 4.2, the user interface consists of two components: 1. A priori-knowledge. 2. Stop conditions.

4.3.1 A priori-knowledge

Let us assume that the user will ask the robot to perform a task with a model free object or environment. Using the proposed system, all what he/she needs is just to define the following information: 1. Sensor configurations. 2. a priori-knowledge about the task (task description). 3. A priori-knowledge about the object (object description), as shown in Fig. 4.3. To make the description easier for the user, it will be suggested a new approach which is convenient to human conception. The robot will be able to take decisions about the control structure and how to perform this task with the help of information provided by sensors, when the user provides the robot only with basic information.

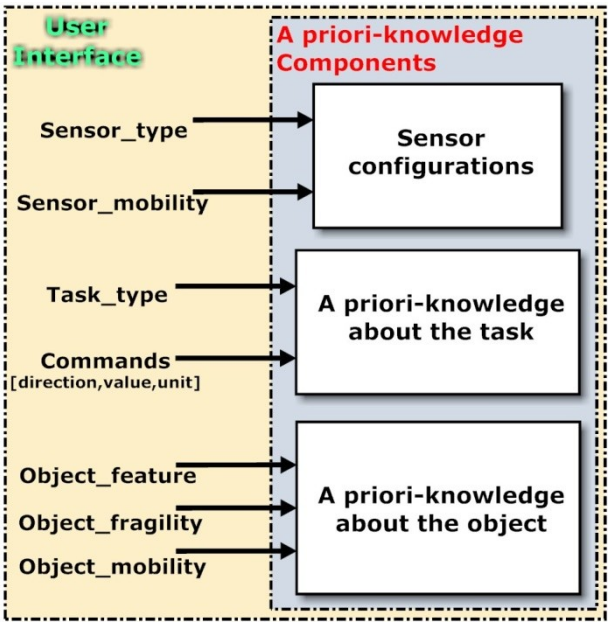

Fig. 4.3 Overview of priori-knowledge components

4.3.1.1 Sensor configurations

As mentioned previously, this work concentrates on vision and force sensor systems. On the one hand, in most applications the force sensor is mounted on robot's wrist in front of the end-effector and it provides information about six components: three Cartesian forces and three Cartesian torques. Consequently, the user doesn't need to provide the robot with the configurations of the force/torque sensor. On the other hand, vision sensor can have many configurations depending on different criteria. Hence, the user will provide the robot with the configurations of the vision sensor by answering the following basic questions:

Is camera 3D? $(\boldsymbol{sensor\_type})$

When the answer is yes, this means that the camera can measure in all directions of the task coordinate system (x_T, y_T, z_T). Otherwise, the used camera is 2D and there is one direction (depth direction) in which the camera cannot measure. Hence, the distance measurement between image to object frame is not possible, especially when the model of the object is unknown, i.e. the vision system can adequately measure maximum 3 characteristics of one feature independently, e.g. feature position (in x and y) and orientation (Θ_z). The purpose of this question is to learn the number of directions and orientations which are able to be vision controlled.

Is camera in hand position? $(\boldsymbol{sensor\_mobility})$

If the answer is yes, this means that in this configuration the camera is mounted on the robot's end-effector, in this case the relationship between the camera coordinate system and the tool coordinate system is constant and represented in ${}^{C}T_{T}$. Hence, sensor velocity is equal to end-effector velocity. This relative position should be determined and it is known as the hand/eye calibration. The position of the target object relative to the camera frame is represented by ${}^{S}T_{C}$ so the position of the target object relative to the robot's end-effector is represented by ${}^{S}T_{T}$ as shown in Fig. 4.4.

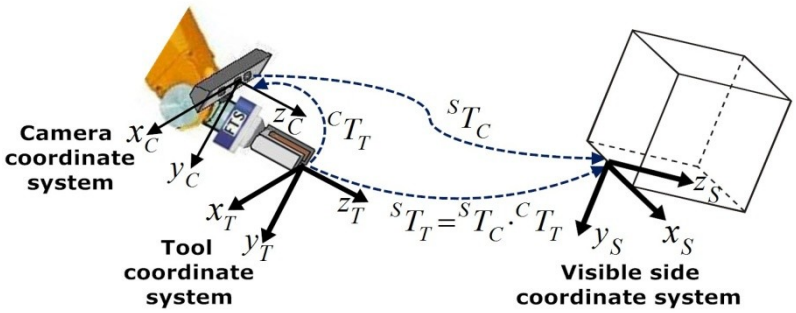

Fig. 4.4 Camera in-hand position

Otherwise, the camera will be mounted in the workspace as shown in Fig. 4.5. Hence, the vision system should be calibrated then the system can calculate the relative position between the camera and the tool coordinate system as follows:

$$\vec{P}_T = {}^{T}\mathrm{T}_0 \cdot {}^{0}\mathrm{T}_C \cdot \vec{P}_C \qquad (4.1)$$

Where (X_0, Y_0, Z_0) is the base coordinate system as shown in Fig. 4.5, ${}^{0}\mathrm{T}_C$ is calculated by camera calibration and ${}^{T}\mathrm{T}_0$ is known by forward kinematics of the robot.

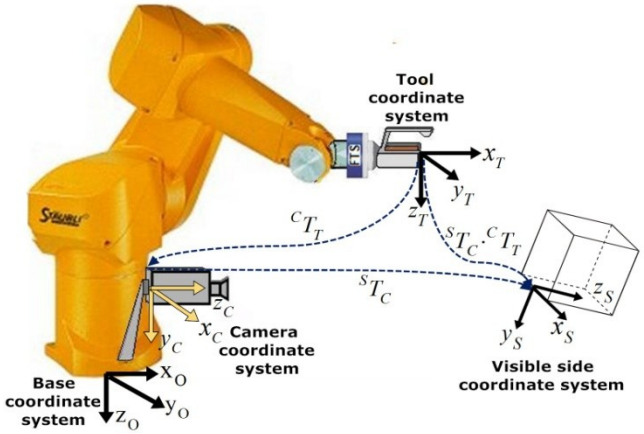

Fig. 4.5 Camera to-hand position

In this case, there are two situations: If the camera has steady position or moving one. In the case of steady position, the camera will be fixed. In the last situation, where the camera is moving the velocity of it should be entered:

$$Sensor\_mobility = \{Configuration, [Direction, Speed]\} \qquad (4.2)$$

where the configuration of the camera will be:

1. In_hand: Here, there is no need for entering the velocity of the sensor, because it is equal to the velocity of the end-effector.
2. To_hand with steady position: Here, the velocity of the sensor will be equal to zero.
3. To_hand with moving position: In this case, the velocity of the camera and its direction should be defined by the user.

The purpose of this basic question (is camera in hand position?) is to find out the relation between tool, camera and the visible side coordinate system. Furthermore, it will help later the system to check if the vision information could be reliably used.

On the whole, when the user only answers these two basic questions with yes or no, the robot system will directly find out which directions can the vision system control and which one can never measure their features. In Fig. 4.5, the vision system cannot measure in direction x_T and in orientations Θ_{Tx}, Θ_{Ty}, if the implemented camera is 2D. In other words, depending on the previous questions, the robot system can also get out if the direction i of the tool coordinate system is parallel to the camera optical axis or not. Here i=1,2,3 indicates the directions of the tool coordinate system.

As shown in Fig. 4.6b, when x_T axis is parallel to the camera main axis and if the camera is 2D, the position of the features perceived by vision system are only in axes z_T and y_T and

orientation Θ_{Tx}. Hence, the vision system can never measure in direction x_T and orientations Θ_{Ty} and Θ_{Tz}. In Fig. 4.6a, the image Jacobian matrix should be defined in order to describe the image feature parameters with respect to the changing of the object or robot pose.

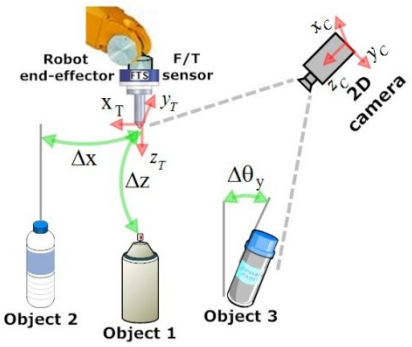

Fig. 4.6a Non-parallel coordinate systems

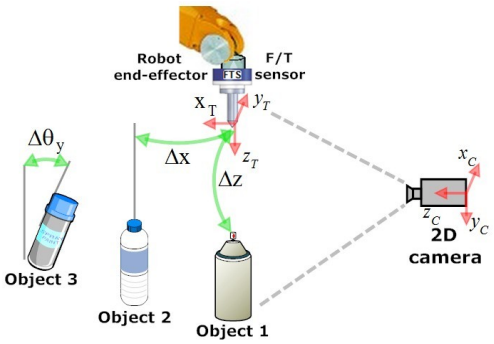

Fig. 1.6b Parallel coodinate systems

4.3.1.2 A priori-knowledge about the task

As shown in Fig. 4.7, the system will propose coordinate system which is convenient to human conception in order to ease process of task description. Human conception coordinate system will be related with world coordinate system and the names of its axes will be convenient to human (right, left, up, etc as shown in Table 4.1 instead of x, y and z). It is known that using the Kinect camera the robot system can easily detect the body of the user, so the robot system will be able to understand the described directions by the user.

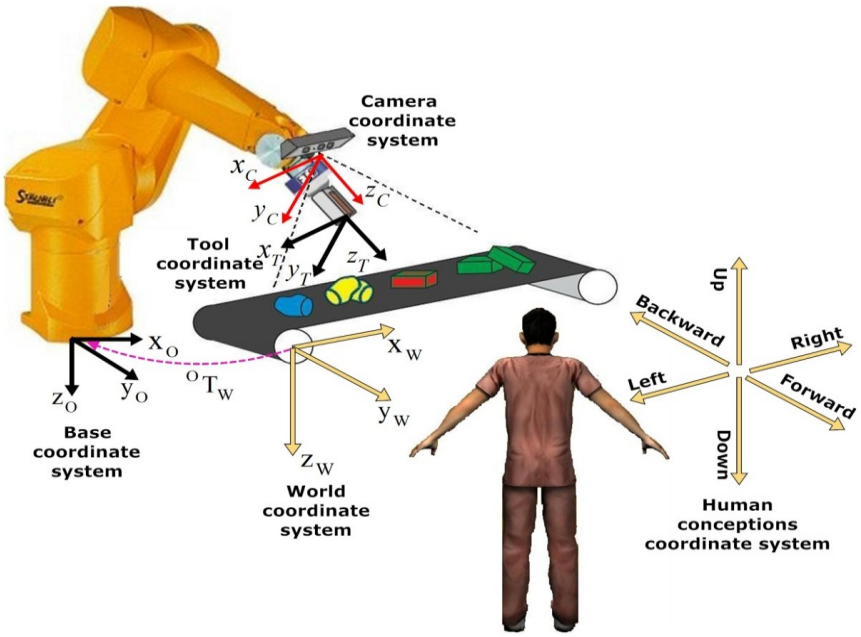

Fig. 4.7 Human conception coordinate system

The world coordinate system will be also related with base coordinate system by transformation matrix ${}^{O}T_{W}$. Hence, the person needs only to enter the following components:

$$Task\_description = \{Task\_type, Desired\_val\} \quad (4.3)$$

This work will classify the robot tasks needing visual servoing into three types:

- Visual servoing with respect to an object for contacting task, e.g. milling, cutting, drilling, polishing, etc.
- Visual servoing with respect to an object for grasping task, e.g. sorting, palletizing, etc.
- Visual servoing with respect to human hand, i.e. transporting objects from/to human hand.

Hence, $Task\_type \in \{contact, grasp, hand\_from, hand\_to\}$, where $Desired\_val$ will define the desired position or force of the proposed task as command signal to the robot system by the user, i.e. $Desired\_val$ represents only the command position and force which are entered by the user.

$$Desired\_val = \{Direction, Value, Unit\} \quad (4.4)$$

where $Unit$ is the physical dimension in SI system. In the proposed system: $Unit \in \{mm\ (position), degree\ (angle), N\ (force), N.m\ (torque)\}$

$Direction \in \{right, left, forward, backward, up, down, up/right, up/left, up/forward, up/backward, right/forward, right/backward\}$. This is simple way for the user to command the robot or to enter the desired values in order to describe the priori-knowledge about the task.

Human conception coordinate system	Desired force/torque in world coordinate system	Desired pose in world coordinate system
Right	$+Fx_W$	$+x_W$
Left	$-Fx_W$	$-x_W$
Forward	$+Fy_W$	$+y_W$
Backward	$-Fy_W$	$-y_W$
Up	$-Fz_W$	$-z_W$
Down	$+Fz_W$	$+z_W$
Up/right	$+Tx_W$	$+\theta x_W$
Up/left	$-Tx_W$	$-\theta x_W$
Up/forward	$+Ty_W$	$+\theta y_W$
Up/backward	$-Ty_W$	$-\theta y_W$
Right/forward	$+Tz_W$	$+\theta z_W$
Right/backward	$-Tz_W$	$-\theta z_W$

Table 4.1 Relation between Human conception and world coordinate system

In general, every task could consist of different subtasks, so the commanded values (either as shown here in task description or later in any command e.g. stop conditions, tracking point, etc.) should be clearly described (if they should be performed in one motion step or they are successive commands). Therefore, the user has possibility to enter ENDSUB command, which means that the subtask is finished and then a new subtask will be performed, e.g. when the user has entered the following:

$$Desired_{val} = \begin{Bmatrix} Right, 100, mm \\ Down, 500, mm \\ Forward, 250, mm \end{Bmatrix}$$

This means that the robot should move in x, z and y as one motion command. In other case, when ENDSUB command is entered as follow:

$$Desired_{val} = \begin{Bmatrix} Right, 100, mm \\ Down, 500, mm \\ ENDSUB; \\ Forward, 250, mm \end{Bmatrix}$$

This means that the robot will move first in x and z direction, and only when this motion is finished then robot will move in y direction.

4.3.1.3 A priori-knowledge about the object

In this section the term object includes also the human hand or even the environment if the robot will perform a task with them. In this work, the poses of the target object is unknown and there is no model of it available. However, the ambition of this work is to perform some interaction tasks between the robot and the target object such as grasping or contacting with the help of vision/force control. The user needs to enter the following information about the target object:

$$Object\_description = \{Obj\_feature, Obj\_fragility, Obj\_mobility\} \quad (4.5)$$

where $Obj\_feature$ could be any features of the target object which help the system to recognize it from the surrounding environment.

Fig. 4.8 Examples of reference images as database

With path/contour following (contact task), $Obj\_feature$ could be e.g. the color of the path.

While grasping an object (grasping task), $Obj\_feature$ could be a single, multi-view images (as shown in Fig. 4.8) or SIFT, SURT features etc. These features are taken by the user or saved previously in the computer system as reference features. In handing-over object from/to human hand, the user needs only to write $Obj\_feature = hand$. This work has implemented two general image processing algorithms for segmentation and detection of any carried object by human hand or any loadfree human hand (see chapter 2). Hence, the user doesn't need to enter any features if the task is the handing-over from/to human hand task.

$Obj\_fragility$ is the fragility factor which defines if the target object can easily be broken or if it has a hard rigidity. This factor will help the robot to define automatically the velocity of motion, especially when it is near from the target object. To find out the optimal velocity of the robot in order to perform the task could be a complicated task for a unprofessional user. In this way, the user needs only to describe how much the object is disposed to be broken as follows:

$$Obj\_fragility \in \{1, \dots, 10\} \tag{4.6}$$

Whenever the object is more tenuous, the fragility factor should be greater and vice versa.

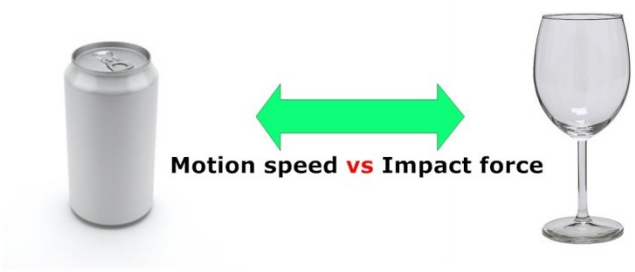

Fig. 4.9 Fragility factor of the target object

As shown in Fig. 4.9, the motion velocity of the robot is related directly with the impact force applied on the target object. In other words, whenever the fragility factor is greater, the motion of the robot should be slower to avoid high impact force especially before establishing the first contact with the target object. In the case of handing-over from/to human hand, the proposed system will assume automatically that $Obj\_fragility = 10$ to guarantee the safety of the user.

On the other hand, $Obj\_mobility$ is the mobility factor which finds out whether the object is fixed or movable and in which direction will it move:

$$Obj\_mobility = \{[Direction, Speed]\} \tag{4.7}$$

If the target object is, e.g., located on a conveyor, the user can enter motion direction of the conveyor relative to the human conception coordinate system and the velocity of the conveyor. This factor will help the system later to analyze if the vision system can reliably detect the object or not and to estimate the pose of the object in the next frame.

4.3.2 Stop conditions

After defining a priori-knowledge about sensor, task and target object, with the help of user interface, the user can provide the system with the stop conditions of the task. The stop conditions will define the termination of the task, i.e. when the robot will finish the task successfully (the robot has arrived at the final conditions, i.e. to desired values) or if an error has occurred (interruption).

$$Stop\_condition = \{Final\_condition, Interrupt\_condition\} \tag{4.8}$$

where $Final\_condition$ is equal to the desired values of the task which are already defined in (4.5) , whereas $Interrupt\_condition$ will define the conditions which will interrupt the task, e.g. when the force values has exceeded a desired value.

$$\begin{gathered} Final\_condition = Desired\_val = \{Direction, Value, Unit\} \\ Interrupt\_condition = \{Direction, Value, Unit\} \end{gathered} \tag{4.9}$$

where $Direction, Value, Unit$ are already explained in equation (4.3). The final status of the robot will depend on stop condition. If the $Final\_condition$ has been activated the robot will signal that the task is successfully finished. Whereas, if the $Interrupt\_condition$ has been activated, this means that task is urgently stopped and it has been performed unsuccessfully.

As shown previously, using the proposed system all the information concerning of sensor description, task description, object description and final condition is facilitated in a simple way that can describe them even if the user is inexperienced. Next, this information will be delivered to the middle interface as the basic information and where it will be combined with the extracted properties of the scene by the vision sensor.

4.4 Scene properties extraction

As shown previously in Fig. 4.2, camera will send the captured image to the scene properties extraction module: In this module the system will analyze the image and it will extract its features in order to define the characteristics of the target object (position, orientation, width, length and etc.) and the characteristics of its surrounding objects. Furthermore, using the vision information the robot will directly calculate the graspability, the tracking point of the object or human hand and it will try to guarantee the safety of the user. Hence, in the proposed system, the robot will not use the vision system only as a simple feedback or as desired position estimator but it will use the vision system to extract the properties of the scene and to find out the surrounding circumstances of the object during performing the task.

The proposed scene properties extraction module consists mainly of three components: 1. Safety of the user. 2. Graspability. 3. Tracking point. It will send one to three components to the middle interface depending on the type of the proposed task, so, in general, the outputs of the scene properties extraction module will be as follows:

$$Props\_extract = \{user\_saftey, graspability, track\_point\} \tag{4.10}$$

If the task is a contact task, e.g. drilling or contour following, the scene properties extraction module will deliver the position of the tracking point or the tracking path (only the third component of equation (4.10)). When the task is grasping, the scene properties extraction module will deliver tracking point of the object and it will calculate the graspability (it will send the last two components of equation (4.10)). However, when task is handing-over from/to the human hand, this module will deliver the three components; user safety, graspability and the tracking point. The next section will illustrate how the vision system can extract all the previous information in order to use them as useful information for defining the control mode.

4.4.1 User safety procedures

In this work, three safety factors are implemented. Two factors are based on vision information and one is based on force information. The first one (SF_body) is related to ensuring the safety of the whole human body depending on the depth map. The second one (SF_hand) is related only to the safety of the fingers during handing-over the object (if the robot is able to grasp the object without touching the human fingers). The third one (SF_force) monitors the force values, especially when the robot is moving toward the human in z direction. Values of these factors will be equal to one as long as the safety requirements are fulfilled. Otherwise, if any error or dangerous position of human is recognized or an unexpected obstacle has encountered the robot, the safety variables will be immediately deactivated, $user\_safety = 0$ and the task will be cancelled. More details about the safety procedures will be illustrated in Chapter 5 provided with some experimental results.

4.4.2 Graspability

This section shows how the robot will calculate the graspability of the object depending on its size and its pose.

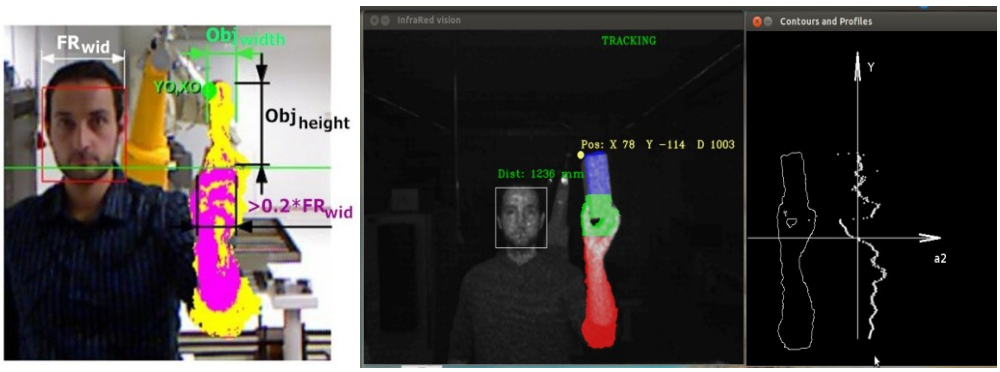

Fig. 4.10 Results of vision algorithms

Fig. 4.10 illustrates the example results of image processing algorithms to detect and segment the target object from the human hand (as shown in Chapter 2). After using the proposed image processing algorithms, the system will define a mask M(i,j), where i = 1,…,640 and j = 1,…,480. Depending on the values of this mask, the pixels of the image will be colored either by yellow (M(i,j)= 1, object) or by violet (M(i,j)=2, human hand). Using this image the robot will

calculate the graspability. Graspability will depend on two different factors. The first one will be related to the object size and the second one will be calculated depending on the pose of the target object and its surrounding objects.

4.4.2.1 Graspability based on object size

As example, when the goal is handing-over object from human hand, the system will define new mask $\widehat{M}_{Hand}(i,j)$. This mask is related only to the human hand and it will be calculated as follows:

$$\widehat{M}_{Hand}(i,j) = \begin{cases} 0 & if \;\; M(i,j) \neq 2 \\ 1 & if \;\; M(i,j) = 2 \end{cases} \tag{4.11}$$

Depending on the mask of human hand $\widehat{M}_{Hand}(i,j)$, the system will find out the boundary line (green line) between the human hand and the object as shown in Fig. 4.11.

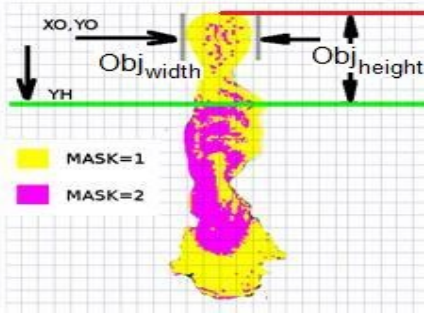

Fig. 4.11 Boundary between human hand and object

The green line illustrates the boundary between the human hand and the object. This boundary will be calculated by scanning every row in the frame (from top to the bottom). If the distance of the pixels which belong to human hand mask ($\widehat{M}_{Hand}(i,j)$) in one row are less than 0.2cm, this row will be considered as segmentation boundary (YH) between the human hand and the object in y direction.

As shown in Fig. 4.12 the graspability will be calculated by comparing width and height of the object ($Obj_{width}, Obj_{height}$) with robot hand width $Grip_{width}$ (distance between the fingers of the gripper) and robot hand height $Grip_{height}$ (height of the robot hand). When $Obj_{width} < Grip_{width}$ this means that the gripper is able to grasp the object. When $Obj_{height} > Grip_{height}$, then the robot is able to grasp successfully. In brief, the object should have a grasp area $GA(i,j)$, where its width is smaller than the width of robot hand and its height is greater than the height of the robot hand in order to activate the $Graspability\_size$ factor:

$$\begin{aligned} &\text{If}((Obj_{height} > Grip_{height}) \;\&\; (Obj_{width} < Grip_{width})) \text{ then} \\ &\qquad \{Graspability\_size \;= 1\} \\ &\text{Else} \qquad \{Graspability\_size \;= 0\} \end{aligned} \tag{4.12}$$

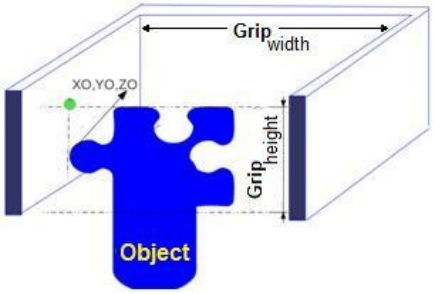

Fig. 4.12 Object and robot hand

If the conditions of graspability based on object size are accomplished, the vision system will complete analyzing the captured image, whereas the robot will cancel the task and it will announce that the task has failed because of the size of the target object.

4.4.2.2 Graspability based on object pose

As mentioned previously, the proposed vision algorithm will not serve only as simple position estimator of the object but it will also extract the relation between the target object and its surrounding objects in order to find out the grasping algorithm (i.e. how the robot will grasp the object with or without force control) and the most appropriated control for every direction (which directions will be force controlled and which vision controlled).

As an example for calculating the graspability of an object depending on its location, this section will present the proposed image processing algorithm to extract the situation of the target book and its neighbor books, to find out if they are stuck together or not. Furthermore, it will find out where/how the robot will be able to grasp the target book in both cases.

To simplify the problem, we will explain at the beginning the relation between a point and a segment, after that we will generalize the procedure to include all cases as shown in Fig. 4.13.

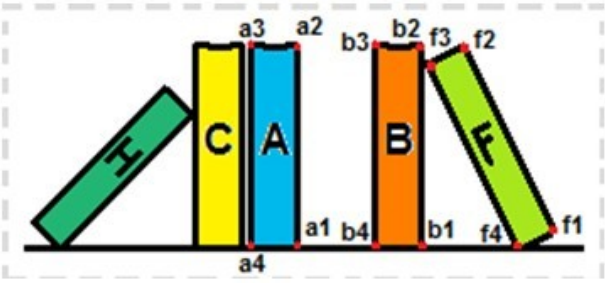

Fig. 4.13 Different situations of neighbor books

As shown in Fig. 4.13, every book has four corners, e.g. corners of book A are:

$$\begin{aligned} \text{Book(A)\_corner} &= [\text{a1}, \text{a2}, \text{a3}, \text{a4}] \\ &= [\text{a}(x1\ y1), \text{a}(x2\ y2), \text{a}(x3\ y3), \text{a}(x4\ y4)] \end{aligned} \tag{4.13}$$

Let us now analyze the relation between one corner of another book, e.g. book B (corner b3) with the segment a1a2 of the book A.

As is well known, the area of the triangle described by three points, in our case (a_{x1}, a_{y1}), (a_{x2}, a_{y2}) and (b_{x3}, b_{y3}), is given by the following equation:

$$T_{area} = \frac{1}{2}\begin{vmatrix} a_{x1} & a_{y1} & 1 \\ a_{x2} & a_{y2} & 1 \\ b_{x3} & b_{y3} & 1 \end{vmatrix} = \tag{4.14}$$

$$\frac{1}{2}(a_{x2} \cdot b_{y3} - a_{y2} \cdot b_{x3} - a_{x1} \cdot b_{y3} + a_{y1} \cdot b_{x3} + a_{x1} \cdot a_{y2} - a_{y1} \cdot a_{x2})$$

T_{area} will be positive if the three points are taken in a anti-clockwise orientation, and negative otherwise.

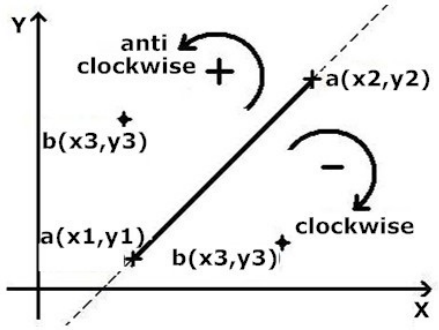

Fig. 4.14 Finding out the sign of area

As shown in Fig. 4.14, points (a_{x1}, a_{y1}) and (a_{x2}, a_{y2}) define the segment. If the point (b_{x3}, b_{y3}) locates on the right side of this segment, which means that the points are taken in clockwise orientation and the value of T_{area} is negative. On the other hand, if the point (b_{x3}, b_{y3}) locates on the left side of the segment so the points are taken in anti-clockwise orientation and the value of T_{area} is positive. If the point (b_{x3}, b_{y3}) locates along the segment, T_{area} will be equal to zero.

On the other hand, as known the area of the triangle can be also calculated using the following equation:

$$T_{area} = \frac{1}{2}\, T_{base} \cdot T_{height} \tag{4.15}$$

where T_{base} is the length of the base of the triangle (in this case the length of segment a_1a_2) and T_{height} is the height of the triangle (in this case the perpendicular distance between point (b_{x3}, b_{y3}) and the segment a_1a_2).

The inclination angle (angle between vertical line and the target book) and the length of the book are calculated in Chapter 2 and T_{area} is already calculated in equation (4.15). Hence, T_{height} can be calculated as follows:

$$norm = T_{height} = 2\frac{T_{area}}{T_{base}} \tag{4.16}$$

We can now calculate the perpendicular distance between any point and any segment even if the segment is vertical or it has inclination angle. The same algorithm will be repeated for all corners of the book B $\mathrm{b}_1, \mathrm{b}_2, \mathrm{b}_3, \mathrm{b}_4$ with the two segments of the book A a_1a_2, a_3a_4 and the results will be as follow:

$$Rel_{BA} = \begin{bmatrix} norm_{b1}^{a1a2}, norm_{b2}^{a1a2}, norm_{b3}^{a1a2}, norm_{b4}^{a1a2} \\ norm_{b1}^{a4a3}, norm_{b2}^{a4a3}, norm_{b3}^{a4a3}, norm_{b4}^{a4a3} \end{bmatrix} \tag{4.17}$$

where Rel_{BA} is the relation matrix between the book B and book A and e.g. $norm_{b1}^{a1a2}$ is the normal distance between the corner b_1 of book B and the segment a_1a_2 of the target book A. From matrix Rel_{BA}, the minimum normal will be calculated which is here $norm_{b3}^{a1a2}$ or $norm_{b4}^{a1a2}$ (they are equal in the case of Fig. 4.13).

The whole algorithm will be repeated twice for every target book: First time with the right side neighbor and the second time with the left side neighbor. In accordance with that, we will have two different cases: either the neighbor books are parallel, like books A and B, or they are not parallel like books B and F.

Parallel neighbors

It can be easily seen that the value of the normal can directly determine if the robot is able to enter its parallel finger between the neighbor books or not. In our case, if $norm_{b3}^{a1a2}$ is greater than the width of robot finger, the space between the book B and book A is large enough to enter the robot finger between them. In contrast, if normal is smaller than the width of robot finger, e.g. $Norm_{c2}^{a4a3}$, the robot will not be able to enter its finger between them. Hence, in our example with book A, the results of the image processing will be as follows: The robot will not be able to grasp the book A in the usual way (entering the parallel fingers round it), therefore another grasping algorithm using the force control will be implemented (it will be illustrated in Chapter 5).

Non parallel neighbors

This case is more complicated than the previous one, see e.g. the pose between book F and book B, book C and book H or even when both neighbor books have inclination angles, such as when books F and H are neighbors.

Fig. 4.15 presents different cases of neighbor books which are not parallel. Lets assume that the target book is the book A and its side which will be analyzed is the green segment a_1a_2. After calculating the normal distance of the corners of the book B to the segment a_1a_2, and using the law of similar triangles:

$$\frac{b_3b_4}{b_4a_1} = \frac{b_3b^{'}}{b^{'}a^{'}} = \frac{b_3b^{''}}{b^{''}a^{''}} \tag{4.18}$$

In this case, the vision system can directly calculate at each point (e.g. $a^{'}$, $a^{''}$) of segment a_1a_2 the length of normal between the target book and its neighbor book as shown in Fig. 4.15

When the normal distance is greater than the width of robot finger, at this point the robot will be able to enter its parallel finger to grasp the target book with no need to use force control.

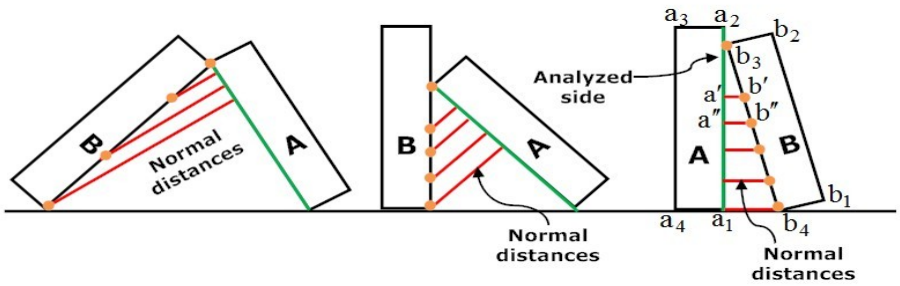

Fig. 4.15 Non parallel neighbor books

On the whole, the same algorithms could be applied in different situations, no matter how the neighbor books are located according to each other or the shape of the target object. Graspability can be described as follows:

$$Graspability = \{Grasp\_method, Grasp\_force\} \tag{4.19}$$

where $Grasp\_method \in \{0, 1, 2\}$ and $Grasp\_force = [Direction, Value, Unit]$.

$Grasp\_method = 0$ means that the robot is not able to grasp the object because either of the conditions of object size or the conditions of objects locations. If the object has a grasp area $GA(i, j)$ which verifies the conditions of graspability based on object size (equation (4.12)) and it is not stuck between other objects (see book B in Fig. 4.13), the robot will grasp the object in the usual way using only the parallel fingers, so $Grasp\_method = 1$ and $Grasp\_force = 0$. Finally, if the grasp area $GA(i, j)$ verifies the conditions of graspability based on object size but it is stuck between other objects (see book A in Fig. 4.13), the robot will try to implement different grasping algorithm with the help of force control by pressing on the object using a third finger. Hence, $Grasp\_method$ will equal 2 and the vision system will learn the direction and the value ($Grasp\_force$) of the pressing force which should be applied on the target book to grasp it. The $Unit$ in (4.19) could be either N or $N.m$.

4.4.3 Tracking point

Tracking point is a point of the target object, contour, human hand etc. which the robot will track and which will be later the first contact point between the object and the robot tool. The robot system has no information about the object's model. Hence, the calculating of the contact point will be performed online and it will be updated within every frame depending on the position of the object or in which way a human is carrying it. Calculating the tracking point will depend on the type of the task and the graspability. In contact task or while transporting object to the human hand, one tracking point is enough, it will be updated within every frame. In the case of grasping task two different algorithms will be proposed:

1. When the target object has no surrounding objects around it and the robot is able to enter its parallel fingers, in this case the robot can grasp the object using only the vision feedback.

Hence, the vision system will calculate the tracking point as follows: Fig. 4.16 illustrates as an example a target object which has somewhat complicated form. Firstly, the robot system will analyze the conditions of the graspability for the object. For example, at point (x_s, y_s), the object width Obj_{width} is greater than the width of robot hand $Grip_{width}$, so the robot is not able to grasp the object in this region. The vision system will scan the whole object. If the object has a grasp area $GA(i,j)$ which verifies graspability conditions as given in equation (4.12) and it is not stuck between two other objects see equation (4.17), the robot system will calculate three points of this area:

a) Upper point (x_U, y_U, z_U),
b) Left point (x_L, y_L, z_L)
c) Nearest point (x_N, y_N, z_N).

The contact point or the tracking point (x_O, y_O, z_O) will be calculated as follows:

$$\begin{aligned} x_O &= x_L \\ y_O &= y_U \\ z_O &= z_N \end{aligned} \tag{4.20}$$

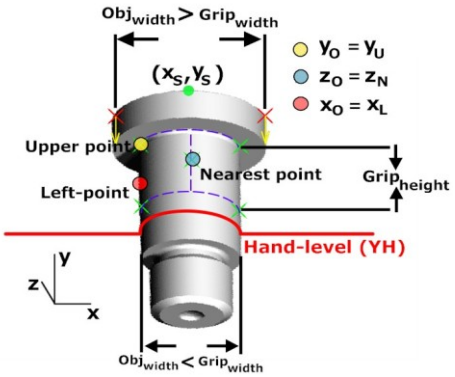

Fig. 4.16 Calculating the contact point

After calculating the contact point of the current frame, the system will use 2D Kalman filter (cvKalman) to filter the noises which could appear in the contact point coordinates. As shown in Chapter 2, the orientation of the robot hand is already calculated, in a way that the two parallel fingers can grip the object.

2. When the target object is stuck between two other objects and there is no sufficient gap to enter the parallel fingers of robot hand. In this case, the vision system will set: a. Tracking point: Where the robot should press on the target object using the third finger in order to rotate the object and to pull it out. b. Force direction: In which direction the pressing force should be applied, as shown in Fig. 4.17. More details about this grasping algorithm will be shown in Chapter 5.

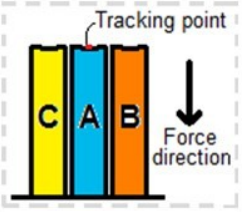

Fig. 4.17 Stucked

$$Track\_point = [Direction, Value, Unit] \tag{4.21}$$

where $Unit$ could be either mm or $degree$.

For better understanding, Fig. 4.18 illustrates the overview of functionality of $Props\_extract$ in different tasks. In the first task (handing-over object from human hand), the first step ensures the safety of the user. When $User\_safety = 0$ (user is unsafe), the system will cancel the task immediately. Otherwise, the system will calculate the graspability. The first phase of graspability will be based on object size (see equation (4.12)): If the conditions are not fulfilled, the task will be canceled. Otherwise, the system will move to check the second level of graspability which is based on object pose (see equation (4.19)). Here, $Grasp\_method$ has three different cases:

1. If the target object is stuck from all directions, the task will be canceled.
2. If the object has no neighbors around it, the system will use only vision feedback and the $track\_point$ will be calculated as shown in equation (4.20).
3. If the object is stuck between two other objects and there is possibility to grasp the object using force control, the system will calculate $track\_point$ and the required force for grasping the object $Grasp\_force$, as shown in Fig. 4.17 and in equation (4.19).

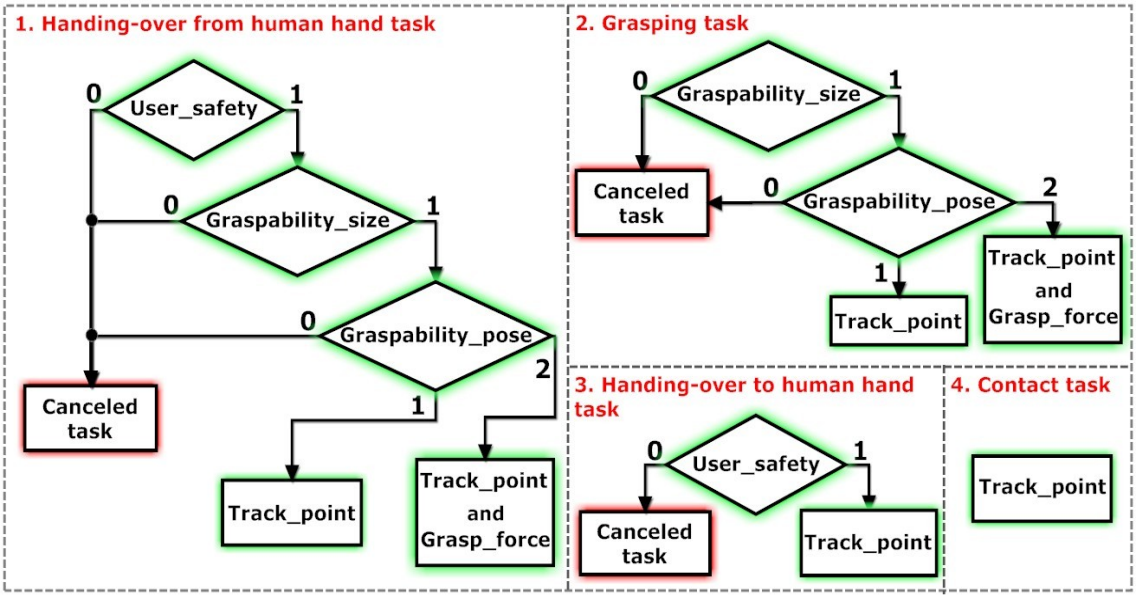

Fig. 4.18 Functionality of $Props\_extract$ in different tasks

In grasping task, the same steps will be followed except the calculation of $User\_safety$, if the robot will not interact with the human. On the other hand, in the tasks where the robot should deliver object to the human hand, the system will calculate $User\_safety$ and $Track\_point$ of the human hand. In the last case (contact task), only $Track\_point$ is required.

4.5 Middle interface

Fig. 4.19 presents an overview of all information which will be analyzed in the middle interface. This information is either supplied by the user as basic information or extracted by the vision system as shown in the previous section. The main task of the middle interface is to analyze all

these information and then combine them together in order to answer the following questions:

- How many directions of the robots should be controlled?
- Which direction should be vision, position or force controlled?
- How to insure that the feedback information could be used reliably?
- How to define what is the most appropriated vision/force control mode?

These questions will be called the pivotal questions. If the proposed automatic decision system can answer these pivotal questions, it will help the robot to perform different successive and complex tasks and to decide automatically the most appropriated combination of vision/force feedback for every task. Furthermore, it will reduce the human intervention or reprogramming during the execution of the task.

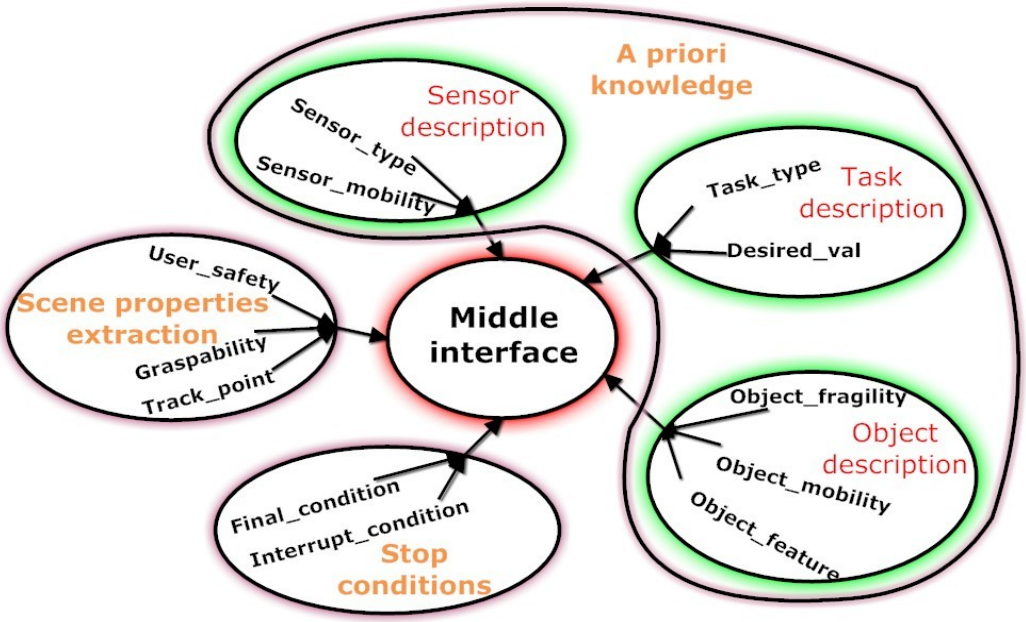

Fig. 4.19 Inputs of middle interface

Next section will illustrate how the robot system can automatically answer the pivotal questions with the help of the basic information provided by the user and the extracted properties of the scene by vision system.

4.5.1 How many directions should be controlled?

In this chapter, we will use the term number of controlled states (NCS) in the task frame, where NCS $\in \{1,2,3,4,5,6\}$. The maximum number of NCS is 6 (three positions and three orientations). The first step is to define the number of controlled subtasks by searching in the task description as previously shown (see equation (4.3)). When the user defines the type of the task, the automatic decision system will count automatically the number of independent

states which should be controlled (NCS) in every subtask to perform the specified task. NCS in the first subtask will be calculated depending on the numbers of command values which are entered by the user in equation (4.4), the number of components in $track\_point$ in equation (4.21) and the number of the components in $Grasp\_force$ in equation (4.19). The values of the interrupt conditions entered in equation (4.9) will be considered as limitation for monitoring the state not as control signal.

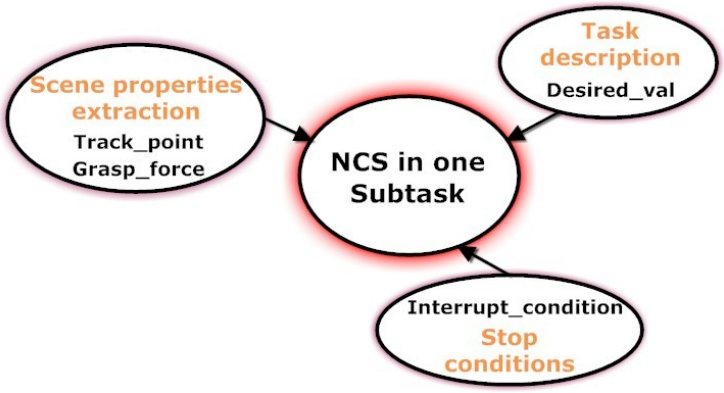

Fig. 4.20 NCS scheme

$$NCS = Num(Desired\_val) + Num(Grasp\_force) + Num(Track\_point) \qquad (4.22)$$

For example in the case when:

$$\text{Desired\_val} = \{Down, 500, mm\}, \qquad \text{Grasp\_force} = \{\ \}, \qquad \text{Track\_point} = \{x, y, \theta_z\}$$

$Num(\text{Desired\_val}) = 1$, $Num(\text{Grasp\_force}) = 0$ and $Num(\text{Track\_point}) = 3$, so the total number of the controlled states in this subtask will be four. Obviously, it is not allowed that one state will be repeated in two different commands for one motion step, because the desired values should come either from the user command or from scene properties extraction.

4.5.2 Which Cartesian direction should be vision, position or force controlled?

After calculating the number of controlled states, the system will define the type of the feedback which will control every independent direction. This work has proposed control frame formalism (CFF), i.e. the control structure in every state. CFF has the same origin of task frame and it defines the control structure in every independent state direction/orientation, i.e. if it should be position, vision or force controlled. For defining CFF, the system will answer the following questions:

- Should position be controlled in this direction?
- Should any force be applied in this direction?
- Is the camera able to measure in this direction?

4.5.2.1 Should position be controlled in this direction?

Robot system can automatically answer this question by looking at $Desired\_val$ in the task description section to know if any position command is entered by the user:

Algorithms 4.1

$For\ \left(i = 1{:}\ Num(Desired\_val)\right)$

$if\ Desired\_val \rightarrow Unit(i) = mm)\ or\ Desired\_val \rightarrow Unit(i) = degree)\ then$

$\mathrm{CCF}(Desired\_val \rightarrow Direction(i)) = \mathrm{Position\_control}$

end

end

where i refers to the independent direction which should be controlled. In $Desired\_val$ if the unit of direction i is measured in mm or in degree, this means this direction will be position controlled. Hence, the matrix value of control frame formalism CFF in that direction will be equal to Position_control.

4.5.2.2 Should any force be applied in this direction?

In the case of force controlled directions, the robot system can automatically answer this question from two matrixes: 1. by looking at $Desired\_val$, if the user has entered any commanded force and 2. by looking at graspability (equation (4.19)), if any force values should be applied to grasp the object.

Algorithm 4.2

$For\ \left(i = 1{:}\ Num(Desired\_val)\right)$

$if\ Desired\_val \rightarrow Unit(i) = N)\ or\ Desired\_val \rightarrow Unit(i) = N.m)\ then$

$\mathrm{CCF}(Desired\_val \rightarrow Direction(i)) = \mathrm{Froce\_control};$

end

end

$For\ \left(j = 1{:}\ Num(\mathrm{Grasp\_force})\right)$

$\mathrm{CCF}(Grasp\_force \rightarrow Direction(i)) = \mathrm{Froce\_control};$

end

The first loop will search for all independent directions which should be controlled by force set point entered by the user. The second loop will also fix the directions which should be force controlled for grasping the object.

4.5.2.3 Is the camera able to measure in certain direction?

This question can be easily answered, as shown previously, after defining $Sensor\_type$, $Sensor\_mobility$ and if the camera main axis is parallel to this direction.

By answering the previous questions, the system will be able to find out if any direction should be position or force controlled and if there is any possibility to combine this controller with the vision information. Because of that, in the next step the robot system will test if the vision information can be reliably used or not.

4.5.3 How to insure that the visual information could be used reliably?

In this section, we will concentrate on the information coming from the vision sensor. If camera can monitor the given direction, the system will test if the image processing results can be reliably used. This testing is carried out by comparing the results of the last captured image with the results of previous captured images. If the difference is not comprehensible and there is no matching between the results of monitoring the robot motion and the updated vision results, the vision cannot be reliably used.

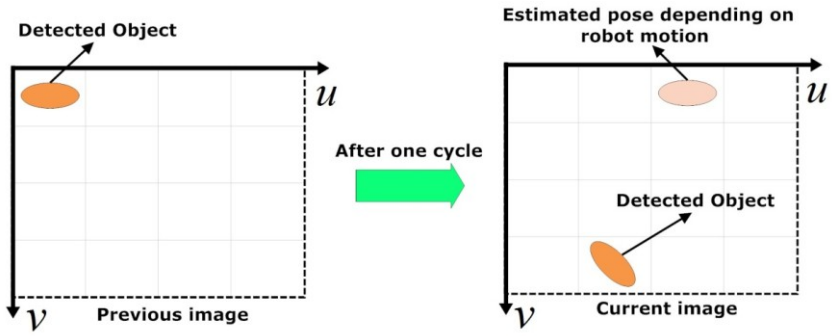

Fig. 4.21 Calculating the reliability factor

This situation can be met in situation as shown in Fig. 4.21, where the robot moves slowly in the direction u only, and when the last two captured images show that the differences of the relative positions of target object with respect to the end effector in all directions have large values. This means that the detection of the target object in one of both images is not correct and the vision feedback cannot be reliably used. Actually, it has no meaning, when the system ignores all vision information because of one or two images with wrong results. In the proposed system if the reliability factor is greater or equal 80% the automatic decision system will activate the vision. The reliability factor will be calculated as follows:

$$\vec{X}_{i+1} = \vec{X}_i + \Delta\vec{O} + \Delta\vec{S} \tag{4.23}$$

where $\vec{X}_{i+1}$ is the estimated pose vector of the object in the frame (i+1), $\vec{X}_i$ the current pose vector of the object in frame (i), $\Delta\vec{O}$ pose offset related to the motion of the object ($Obj\_mobility$ as shown in equation (4.7)) and $\Delta\vec{S}$ pose offset related to the motion of the sensor ($Sensor\_mobility$). The reliability factor will be calculated for every 10 frames by comparing the estimated pose $\vec{\hat{X}}_{i+1}$ and the measured pose $\vec{X}_{i+1}$. If the vision information is completely wrong in maximum two frames, the vision system will be activated and it can be reliably used.

This algorithm cannot be implemented when the object is moving and its velocity is unknown, e.g. in the case of handing-over object from human hand. In this case, the human hand carries the target object, it can move randomly and with variable velocity and acceleration, so the robot system cannot estimate the position of the object in the next frame. However, the proposed visual algorithms in this work for handing-over tasks have shown a high efficiency during the experiments and no error has occurred for detecting the loadfree human hand and also for the object carried by human hand.

4.5.4 How to define the most appropriated vision/force control mode?

In the first chapter, the structures of the vision/force control i.e. shared, traded and hybrid are illustrated according to (Nelson, Morrow, & Khosla, 1996). However, if we assume that $P_T(x_T, y_T, z_T)$ is the position of the point P of the target object relative to task coordinate system, the automatic decision algorithm will set the structure for fusing vision and force control in all directions of the task coordinate system (X_T, Y_T, Z_T) separately and depending on the following definitions of the vision/force control structure, i.e. we will redefine the structures of vision/force control as follows:

Vision/force control structure	Definition
Pure position	When camera cannot measure in this direction and the force shouldn't be applied but there is a commanded position.
Pure vision	When camera can reliably measure in this direction, the force shouldn't be applied and there is no commanded position.
Pure force	When camera cannot measure in this direction, a desired force should be applied and there is no commanded position.
Traded control	When camera can reliably measure in this direction, a desired force should be applied and there is no commanded position.
Shared control	When camera cannot reliably measure in this direction, a desired force should be applied and there is no commanded position.
Vision/guarded-force control	When camera cannot reliably measure in this direction, the force shouldn't be applied and there is no commanded position.
Hybrid control	When camera can reliably measure in this direction and there is no commanded position in this direction but a desired force should be applied in the orthogonal directions to this direction.

Table 4.2 Definitions of vision/force control structures

Depending on the previous definitions, the automatic decision system can define the appropriated vision/force robot control in order to benefit from all the advantages of different control structure. Fig. 4.22 shows the main part of automatic decision algorithm which finds out the structure of fusing vision and force control in all directions x, y and z. This algorithm

will be repeated for every direction x, y and z. At first the system will test if the direction is parallel to the camera axis, which would mean that camera cannot measure in this direction unless it is 3D camera. So if the camera is 2D and parallel to this direction, the vision sensor cannot be used in this direction. After that if there is a desired force in this direction, the control mode is force control, i.e. $S_{Pi} = 0$, $S_{Vi} = 0$ and $S_{Fi} = 1$. Otherwise, when no force is applied, this direction will be command position controlled, i.e.$S_{Pi} = 1$, $S_{Vi} = 0$ and $S_{Fi} = 0$.

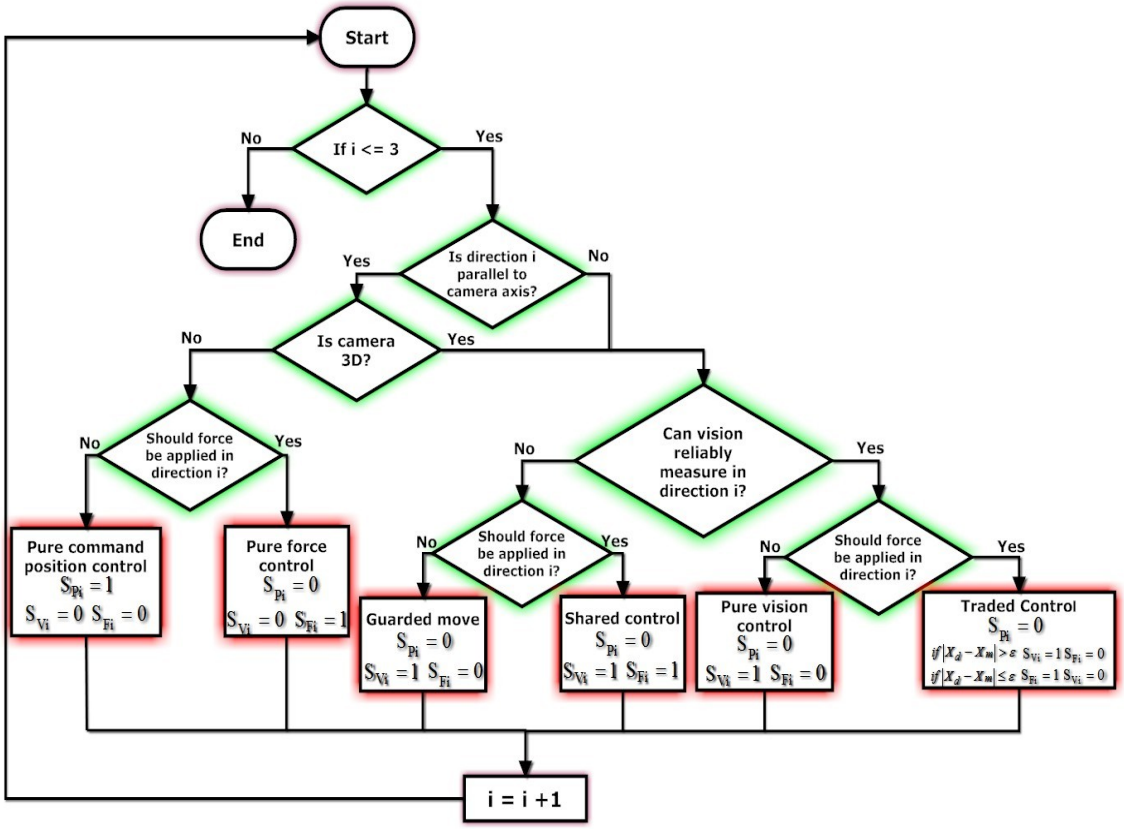

Fig. 4.22 Automatic decision algorithm for directions x, y, z

On the contrary, when the camera can monitor this direction, the system will test if the image processing results can be reliably used. If the difference is not comprehensible and there is no matching between the results of monitoring the robot motion and the updated vision results, the vision cannot be reliably used. This situation can be met in situation, when the robot moves slowly in the direction x only, and when the last two captured images show that the differences of the relative positions of target object with respect to the end-effector in all directions have large values. This means, that the detection of the target object in one of both images is not correct and the vision information cannot be reliably used. In this case, if there is a given desired force in this direction, the control mode will be shared control, i.e. $S_{Pi} = 0$, $S_{Vi} = 1$ and $S_{Fi} = 1$. However, if there is no given desired force, this direction will be vision/guarded-force controlled ($S_{Pi} = 0$, $S_{Vi} = 1$ and $S_{Fi} = 0$), i.e. the force sensor will be

monitored. Using the vision information in the last two situations will depend on the reliability factor as shown previously in equation (4.23).

On the other hand if the vision information can be reliably used and if there is no given desired force in this direction, the control mode will be vision control, i.e. $S_{Pi} = 0,\ S_{Vi} = 1$ and $S_{Fi} = 0$. However, if there is a given desired force in this direction and vision information can be reliably used, the control mode will be traded control, as equations (4.24) and (4.25) show:

$$\text{if } |X_d - X_m| > \varepsilon \ \ then\ S_{Pi} = 0, S_{Vi} = 1,\ S_{Fi} = 0 \qquad (4.24)$$

$$\text{if } |X_d - X_m| \leq \varepsilon \ \text{ then} S_{Pi} = 0, S_{Vi} = 0,\ S_{Fi} = 1 \qquad (4.25)$$

Here ε is a small threshold to switch from vision feedback to force feedback. The system will define ε value depending on the priorities of impact quality and motion velocity.

Fig. 4.23 shows the main part of automatic decision algorithm which sets the structure for combining vision and force control in orientation angle θ_z depending on the control structure of x and y directions. The same algorithm will then be applied to the other angles (θ_x depending on y and z and θ_y depending on x and z).

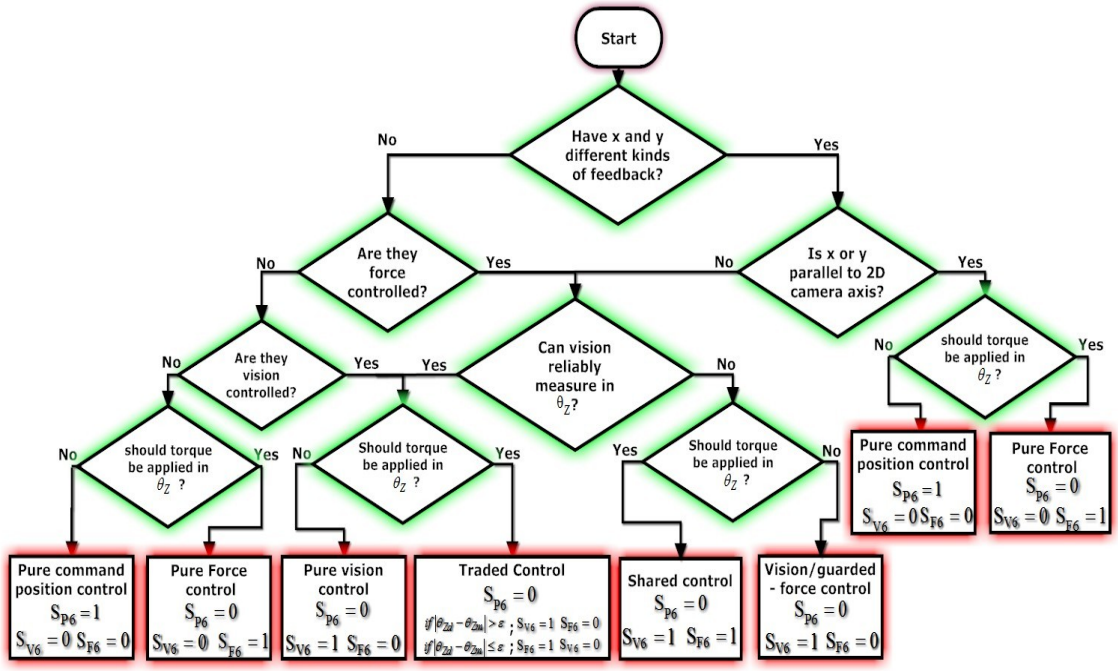

Fig. 4.23 Automatic decision algorithm of orientation angle

The explanation of the relation between the control structure in x, y and the control structure in θ_z is as follow: 1. For vision control: If both directions x and y are vision controlled, i.e. the camera can be reliably used in x and y direction. Hence, the robot system can automatically decide that the vision system can also be reliably used in θ_z. 2. For force control, usually force/torque sensor is mounted between the last joint of the robot and the gripper i.e. in the

robot wrist, so the coordinate system of the force/torque sensor is shifted from the tool coordinate system. This means that if the robot has a contact force at one point in the gripper, this force will produce a torque in the sensor coordinate system. If the contact forces between the gripper and the object are desired along x and y, the torque value in the force/torque sensor coordinate system could also be measured about z axis, so θ_z will be force controlled to insure better quality of impact.

In Fig. 4.23, The pure force or position control could be implemented in two different situations: 1. When x and y directions are not vision or force controlled and there is a given desired torque or commanded position in θ_z. 2. If x and y directions have different control structures and one of them is parallel to the optical axis of the camera (camera cannot measure the features in this orientation θ_z). In both these cases, if there is a desired torque in this orientation, the control mode is force control, i.e. $S_{V6} = 0$, $S_{P6} = 0$ and $S_{F6} = 1$, whereas this orientation will be position controlled ($S_{V6} = 0$, $S_{P6} = 1$ and $S_{F6} = 0$).

The pure vision control and traded control could be implemented also in two different situations: 1. Both x and y axes are vision controlled, i.e. the camera can be reliably used in x and y direction. Hence, the robot system can automatically decide that the vision system can also be reliably used in θ_z. 2. At least one of both axes is force controlled and the camera can reliably measure the feature in this orientation. In this case, the automatic decision system needs to insure that the camera can measure in this orientation reliably and to calculate the reliability factor. In both cases, the control mode will be as follows: If there is no desired torque in θ_z, the control mode will be pure vision control, i.e. $S_{V6} = 1$, $S_{P6} = 0$ and $S_{F6} = 0$. Otherwise, it will be traded control:

$$\text{if } |\theta_{zd} - \theta_{zm}| > \varepsilon \ \textit{then} \ S_{P6} = 0, S_{V6} = 1, S_{F6} = 0 \qquad (4.26)$$

$$\text{if } |\theta_{zd} - \theta_{zm}| \leq \varepsilon \ \text{then} \ S_{P6} = 0, S_{V6} = 0, S_{F6} = 1 \qquad (4.27)$$

On the other hand, where the camera cannot be reliably used in θ_z, here the control mode will be shared control, i.e. $S_{P6} = 0, S_{V6} = 1$ and $S_{F6} = 1$, when a given desired torque is applied in this orientation. Otherwise, this orientation will be vision/guarded-force controlled, i.e. the force sensor is only monitored ($S_{P6} = 0, S_{V6} = 1$ and $S_{F6} = 0$).

In Fig. 4.22 and Fig. 4.23, the hybrid control is not illustrated explicitly. The previous algorithms are designed for every single direction or orientation only. On the other hand, hybrid control works in the whole space. Hence, once can see the hybrid control obviously only from the selection matrices S_P, S_V and S_F.

In conclusion, the proposed system will analyze the situation of the task and the conditions of vision and force information in every direction in order to use all the advantages of different vision/force control structures and to avoid disadvantages.

4.6 Control system structure

This section will discuss the structure of the proposed control system and the algorithms of fusing vision and force control. Fig. 4.24 shows the scheme of control system which consists of four blocks: automatic decision system, environment, position loop and force loop.

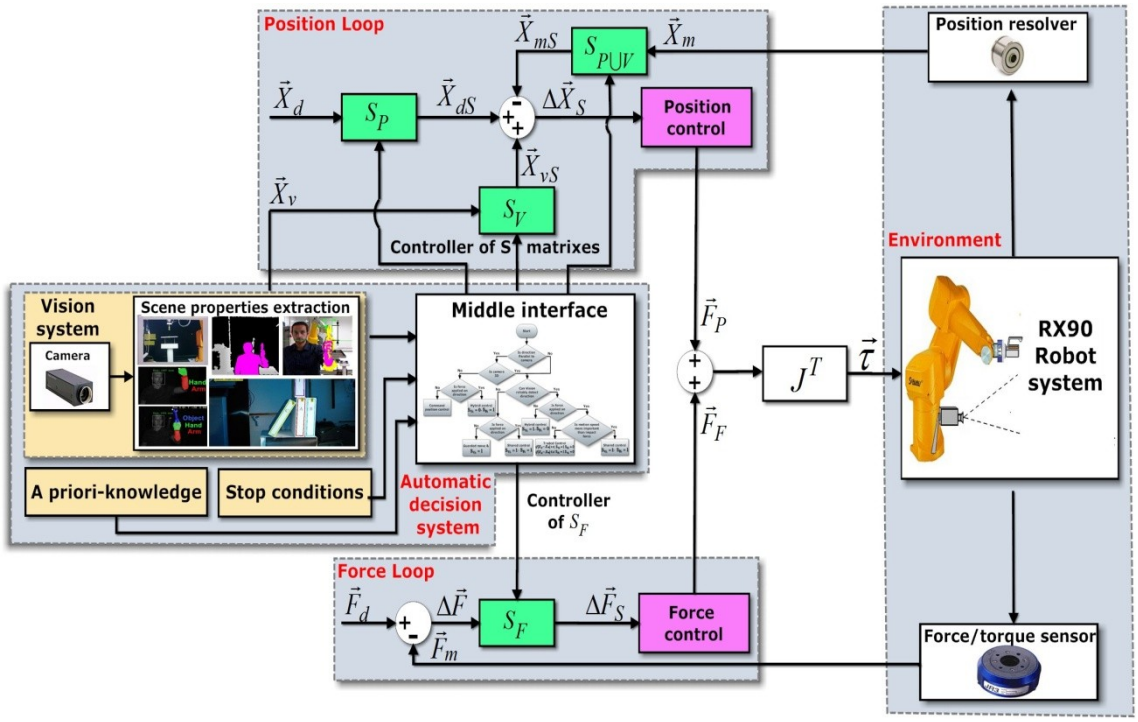

Fig. 4.24 Scheme of control system

Vision system consists of camera, scene properties extraction module and automatic decision module as shown previously. Camera sends the captured image to the scene properties extraction module. As shown previously, in this module the system will analyze the image and extract the properties of the scene. After that the scene properties extraction module will send the pose of the target objects $\vec{X}_v$ to the position loop and the extracted features of the target object and surrounding objects to automatic decision module. As shown previously, the automatic decision module will do the following:

a) Analyzing the conditions of the tasks for every direction.
b) Analyzing a priori knowledge of the sensor and the target object.
c) Calculating the graspability depending on the object size and on the relative pose of target object with respect to the surrounding objects and environment.
d) Testing the reliability of using vision and force information for every direction.
e) Checking the values of the selection matrixes.

All these operations are performed in order to define the contacting or grasping algorithm and to specify the most appropriate combination structure of vision/force control.

Hence, the last step in the proposed system is to check the values of selection matrixes S_P, S_V and S_F, where S_P, S_V and S_F are position, vision and force selection matrixes. Selection matrices are diagonal and their entries are either 1 or 0:

$$S_P = \begin{bmatrix} S_{P1} & 0 & 0 & 0 & 0 & 0 \\ 0 & S_{P2} & 0 & 0 & 0 & 0 \\ 0 & 0 & S_{P3} & 0 & 0 & 0 \\ 0 & 0 & 0 & S_{P4} & 0 & 0 \\ 0 & 0 & 0 & 0 & S_{P5} & 0 \\ 0 & 0 & 0 & 0 & 0 & S_{P6} \end{bmatrix} \quad (4.28)$$

$$S_V = \begin{bmatrix} S_{V1} & 0 & 0 & 0 & 0 & 0 \\ 0 & S_{V2} & 0 & 0 & 0 & 0 \\ 0 & 0 & S_{V3} & 0 & 0 & 0 \\ 0 & 0 & 0 & S_{V4} & 0 & 0 \\ 0 & 0 & 0 & 0 & S_{V5} & 0 \\ 0 & 0 & 0 & 0 & 0 & S_{V6} \end{bmatrix} \quad (4.29)$$

$$S_F = \begin{bmatrix} S_{F1} & 0 & 0 & 0 & 0 & 0 \\ 0 & S_{F2} & 0 & 0 & 0 & 0 \\ 0 & 0 & S_{F3} & 0 & 0 & 0 \\ 0 & 0 & 0 & S_{F4} & 0 & 0 \\ 0 & 0 & 0 & 0 & S_{F5} & 0 \\ 0 & 0 & 0 & 0 & 0 & S_{F6} \end{bmatrix} \quad (4.30)$$

As shown previously, with the help of automatic decision module, the system will decid automatically which directions will be force controlled and which position or vision controlled and when the system will switch from one mode to the other. In the force loop the system will calculate force error vector $\Delta\vec{F}\,(6\mathrm{x}1)$ from the desired force vector $\vec{\mathrm{F}}_\mathrm{d}$ and measured force vector $\vec{\mathrm{F}}_\mathrm{m}$:

$$\Delta\vec{F} = \vec{F}_d - \vec{F}_m \quad (4.31)$$

After that the system will apply the force control only in the directions which should be force controlled by multiplying the force error vector $\Delta\vec{\mathrm{F}}$ with force selection matrix $\mathrm{S_F}$:

$$\Delta\vec{F}_S = S_F \cdot \Delta\vec{F} \quad (4.32)$$

The output of force control is $\vec{\mathrm{F}}_\mathrm{F}$, which is the force/torque vector corresponding to the selected force/torque error.

In the position loop: With $\vec{\mathrm{X}}^\mathrm{T} = [\mathrm{X}, \mathrm{Y}, \mathrm{Z}, \Theta_\mathrm{X}, \Theta_\mathrm{Y}, \Theta_\mathrm{Z},]$ will be further denoted the pose vector of end-effector. With indices m, d and v will be denoted the sources of pose vector $\vec{\mathrm{X}}$: measured, commanded or vision. The automatic decision module will find out which directions can reliably be vision controlled and which will be controlled with desired position $\vec{\mathrm{X}}_\mathrm{d}$ by using the matrices $\mathrm{S_P}$ and $\mathrm{S_V}$. For the position control loop, the system will calculate $\vec{\mathrm{X}}_\mathrm{dS}$ and $\vec{\mathrm{X}}_\mathrm{vS}$ which determine the directions which will be vision or position controlled (see Fig. 4.24):

$$\vec{X}_{dS} = S_P \cdot \vec{X}_d \quad (4.33)$$

$$\vec{X}_{vS} = S_V \cdot \vec{X}_v \quad (4.34)$$

Also the measured pose $\vec{X}_m$ will be multiplied with the sum of S_P and S_V, because the position control will be applied only in the directions which are controlled with vector $\vec{X}_v$ (desired position from vision) or with commanded position vector $\vec{X}_d$:

$$\vec{X}_{mS} = (S_P + S_V) \cdot \vec{X}_m \quad (4.35)$$

At the end, the system will calculate pose error vector $\Delta\vec{X}_S$ as follows:

$$\Delta\vec{X}_S = \vec{X}_{dS} + \vec{X}_{vS} - \vec{X}_{mS} \quad (4.36)$$

In the proposed system there is no relation between S_V and S_F, in other words ($S_V \neq (1 - S_F)$) as it used in the hybrid control. This proposed system includes all possibilities of combination vision and force and with help of automatic decision module.

If the automatic system has refused the control task, the selection matrix automatically tries to choose an alternative control method. Should no suitable one be available, the robot will cancel the task or return home. For example, if the task is to grasp a book from bookrack, the first controller will lead the robot to grasp the robot using the parallel fingers while using only the visual servoing control loop. However, if the target book is stuck between two other books, an alternative controller will be implemented which combines vision and force control loop using three fingers. In the worst case, if the target book is stuck from all directions, robot will declare that the proposed task can't be performed.

4.7 Conclusion

This chapter has suggested an automatic decision system which decides automatically about the most appropriate vision/force control structure for different tasks depending on the surrounding environment and the preconditions of tasks. This work has used all possible types of vision/force control combinations and it could use different structures of the control in different directions in the same task to insure good quality of control. This strategy will allow the robot to benefit from all the advantages of different control structures and to perform the complex tasks with no need to be re-programmed or intervened by human. This chapter has divided the tasks requiring visual servoing into three types: 1. Visual servoing with respect to an object for contacting task. 2. Visual servoing with respect to an object for grasping task. 3. Visual servoing with respect to human hand. Furthermore, a user interface has been proposed. It consists of basic questions which any user can answer before starting the task. The user interface has made the description easier for the user. In addition to that, this chapter has illustrated how the robot system can benefit from all the available information which could be provided by the sensor, not only for the target object but also for the whole scene. The practical implementation of the proposed system will be illustrated in the next chapter.

Chapter 5

Practical Implementations of Vision/ Force Integration

Integration of vision and force control has many advantages in the robot applications and usually it is used to get better performance in different tasks, especially by implementing different structures of vision/force control. The exploitation of these advantages is the key contribution of this chapter. For this purpose, this chapter will introduce different scenarios illustrating clearly the benefits of vision/force integration in different robot applications.

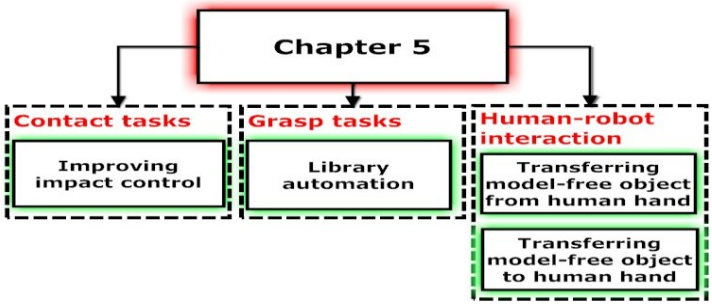

Fig. 5.1 Overview of Chapter 5

As previously mentioned we have classified robot tasks needing visual servoing into three types:

1. Visual servoing with respect to an object for contacting task, e.g. milling, cutting, drilling etc. Our scenario for this kind of robot application will illustrate how the system can improve the impact control with the help of vision/force integration.
2. Visual servoing with respect to an object for grasping task, such as sorting and palletizing systems. The chosen scenario in this category is an automated sorting system in libraries. This scenario will present the facilities of vision/force integration in order to grasp imprecisely objects with different poses and grasping situations.
3. Visual servoing with respect to human hand. Here, the scenario is transferring model-free objects between human hand and robot hand. It will show how the human robot interaction could be improved by fusing vision and force control.

All these scenarios will be described in the following sections and they will be supported with experimental results.

5.1 Vision/force integration for contacting task

In the most contact tasks the robot motion must change between free space motion and the constrained motion. Usually, switching from constrained force control to unconstrained position control presents no problems. However, switching from free space motion to constrained force control generally causes impact forces which can be very large. A large impact force can drive an otherwise stable control loop into instability and it can be harmful to the manipulator hardware or to the environment. Therefore, this section will illustrate how to improve the impact control using vision/force robot control experimentally.

5.1.1 Improving impact control with the help of vision/force control

The concentration of the impact forces in this work has leaded to propose this scenario: the robot will move in only one axis (z axis) toward the environment which is a spring loaded table. The vision system will observe the movement of the robot end effector and calculate the distance between it and the work table. After that, the robot system will reduce its speed according this distance. The performance of the robot will be presented in two cases, with and without vision information. This section will present the hardware, robot controller and software of the experiment after that the simulation and experimental results will be illustrated.

5.1.1.1 Experimental equipments

Fig. 5.2 presents the overall experimental setup which consists of:

① Stäubli RX90 robot.
② FT-Delta SI-660-60 force/torque sensor which is produced by SCHUNK. The effective measurement range of the force/torque sensor used is ±660 N for forces and 60 Nm for torques.
③ Sony DFW-X700 color digital camera is mounted in camera to Hand configuration. Its frame rate is 15 frames/s and the transfer rate is 400 Mbps with IEEE 1394 digital interface.
④ Environment is designed as spring-loaded table.
⑤ Tool is designed as finger in order to fit the purpose of this experiment. The end-effector is installed on the collision protection device.
⑥ Three markers are installed (green circles), two on the finger and one on the contact surface.

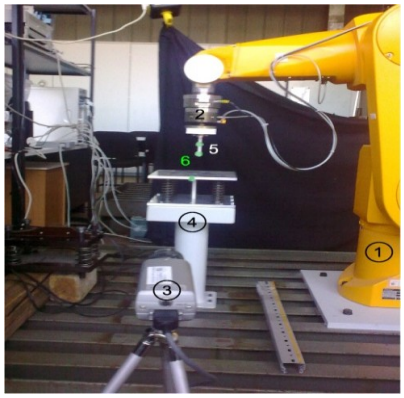

Fig. 5.2 Hardware

Fig. 5.3 demonstrates the structure of the robot system and environment components. The environment is actually spring loaded table which consists of base, four springs, and the contact surface which is a metal board mounted on the springs. Using the green markers the system will calculate the exact position of the end-effector relative to the spring loaded table.

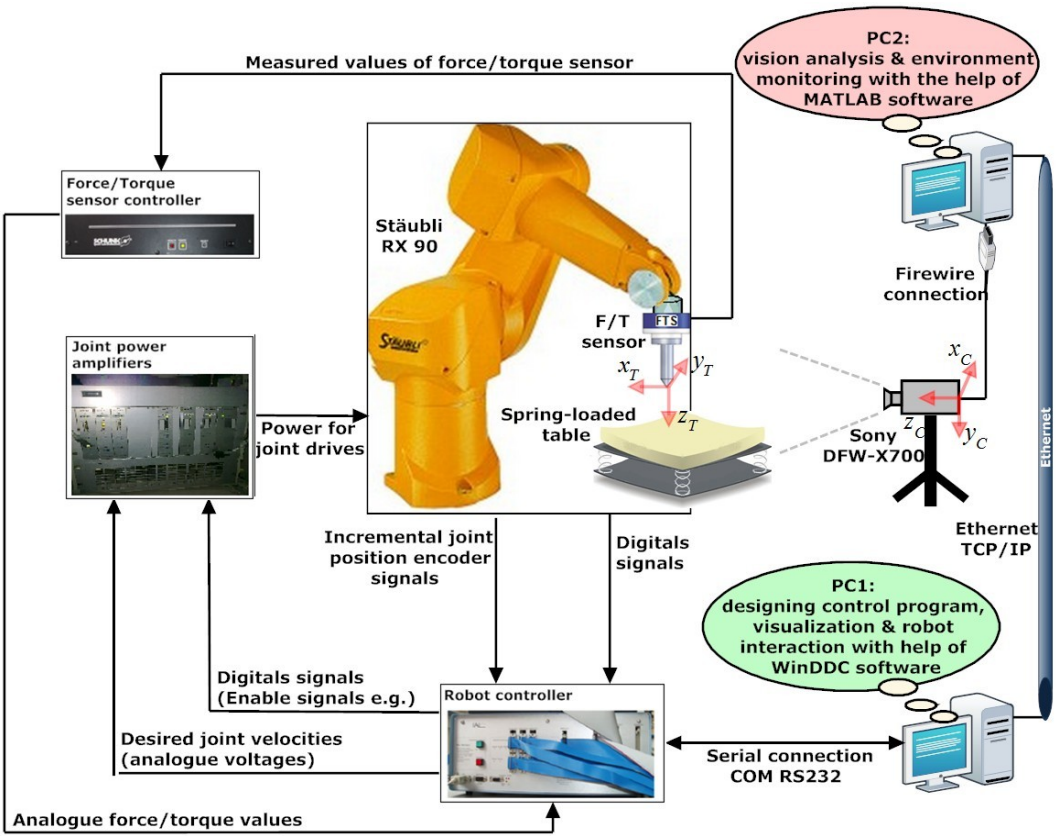

Fig. 5.3 Overview of hardware equipments

5.1.1.2 Robot controller

Improved impact control and development of force and vision algorithms was preceded by the development of an adequate control system. This was unavoidable as control systems of common commercial robots do not still admit good working environment. The interpolation time of the new robot controller is 1 ms and it can be reduced to 500µs. From the original robot control system just the joint power amplifiers have been preserved. Each power amplifier includes motor current controller. The motor current can be controlled by analogue voltages (±10 V for maximum current in positive/negative direction). The actual joint angles are measured by incremental position encoders. The new robot controller, WinDDC-Real-Time-Controller (Winkler & Suchý, 2005), is based on Analog Devices ADSP 2181 digital signal processor. Robot controller is equipped with digital and analogue inputs and outputs, inputs for incremental position encoders, interface for industrial field busses (CANBus, e.g.) and serial interfaces. Robot controller is programmed by special language called DDC (Neumann, 1991). This programming language contains commands enabling access to the peripherals and to software elements of control technology like integrators, differentiators etc. The program development is performed with a standard PC (Windows operating system) and WinDDC software. It is possible to keep influence on program variables during execution by PC and supervise these variables from the robot controller by their visualization. In addition to that PC simulations are also possible with WinDDC.

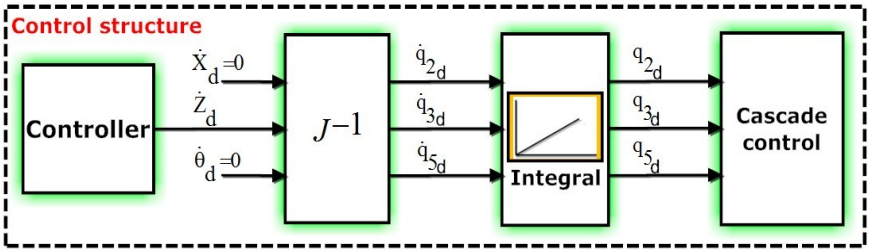

Fig. 5.4 Control structure of the experiment

As previously mentioned, from the original robot control system just the joint power amplifiers have been preserved. Each power amplifier includes the processing of measured signals of the resolver and motor current controller. The proposed controller provides the desired Cartesian values of end-effector speed as output which has an advantage that the desired values of the position and orientation of the end-effector can be generated. Hence, the controller can be implemented in the future in commercial robot controllers, if the appropriated interface in reference (Winkler & Suchý, 2006) is available. For these reasons, it has chosen a decentralized joints control in the new robot control with a classic cascade control structure as shown in Fig. 5.4. The DDC controller controls the speed and position of all 6 robot joints. The speed controller is PI controller and the position controller is proportional controller. Furthermore, for every joint there is velocity feedforward controller in order to minimize the tracking error.

With the help of inverse Jacobi matrix J^{-1}, the program will convert the Cartesian desired velocity $\vec{X}_d$ to the corresponding joints velocity $\vec{q}_d$ as follows:

$$\vec{q}_d = J^{-1} \cdot \vec{X}_d \tag{5.1}$$

In this scenario the manipulator will work only in the x-z plane of the world coordinate system. In addition to that all calculation of forward/backward transformation will be performed manually because of using the new controller system. Hence, the size of the forward transformation and the Jacobi matrix will be reduced. The controller system will control the joints q_2, q_3 and q_5, whereas the rest joints will be equaled to zero $q_1 = 0$, $q_4 = 0$ and $q_6 = 0$. Hence the forward transformation will be:

$$^{0}\underline{A}_{3} = \begin{bmatrix} C23 & 0 & S23 & a_2C2 \\ 0 & 1 & 0 & 0 \\ -S23 & 0 & C23 & -a_2S2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.2}$$

$$^{3}\underline{A}_{6} = \begin{bmatrix} C5 & 0 & S5 & d_6S5 \\ 0 & 1 & 0 & 0 \\ -S5 & 0 & C5 & d_4 + d_6C5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.3}$$

From (5.2) and (5.3):

$$^{0}\underline{A}_{6} = \begin{bmatrix} C23C5 - S23S5 & 0 & C23S5 + S23C5 & d_6C23S5 + d_4S23 + d_6S23C5 + a_2C2 \\ 0 & 1 & 0 & 0 \\ -S23C5 - C23S5 & 0 & -S23S5 + C23C5 & -d_6S23S5 + d_4C23 + d_6C23C5 - a_2S2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.4}$$

As is known the transformation matrix T is written as follows:

$$T = \left[\begin{array}{ccc:c} r_{11} & r_{12} & r_{13} & p_X \\ r_{21} & r_{22} & r_{23} & p_Y \\ r_{31} & r_{32} & r_{33} & p_Z \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \tag{5.5}$$

From (5.4) and (5.5) the position and orientation vector can be written:

$$\vec{P} = \begin{bmatrix} x \\ z \\ \phi \end{bmatrix} = \begin{bmatrix} d_6C23S5 + d_4S23 + d_6S23C5 + a_2C2 \\ -d_6S23S5 + d_4C23 + d_6C23C5 - a_2S2 \\ q_2 + q_3 + q_5 \end{bmatrix} \tag{5.6}$$

As is known, the equation of Jacobi matrix can be written as follows:

$$\underline{J} = \begin{bmatrix} \frac{dx}{dq_2} & \frac{dx}{dq_3} & \frac{dx}{dq_5} \\ \frac{dz}{dq_2} & \frac{dz}{dq_3} & \frac{dz}{dq_5} \\ \frac{d\phi}{dq_2} & \frac{d\phi}{dq_3} & \frac{d\phi}{dq_5} \end{bmatrix} \tag{5.7}$$

Hence, using (5.6) and (5.7) the 3x3 Jacobi matrix will be:

$$J = \begin{bmatrix} -d_6S23S5 + d_4C23 + d_6C23C5 - a_2S2 & -d_6S23S5 + d_4C23 + d_6C23C5 & d_6C23C5 - d_6S23S5 \\ -d_6C23S5 - d_4S23 - d_6S23C5 - a_2C2 & -d_6C23S5 - d_4S23 - d_6S23C5 & -d_6S23C5 - d_6C23S5 \\ 1 & 1 & 1 \end{bmatrix} \tag{5.8}$$

The inverse Jacobi matrix will be calculated by the determinant of the matrix as follows:

$$\underline{J}^{-1} = \frac{\mathrm{adj}(\underline{J})}{\det(\underline{J})} \tag{5.9}$$

Where,

$$\det(\underline{J}) = J_{11}J_{22} - J_{11}J_{23} - J_{12}J_{21} + J_{12}J_{23} + J_{13}J_{21} - J_{13}J_{22} \tag{5.10}$$

$$\mathrm{adj}(\underline{J}) = \begin{bmatrix} J_{22} - J_{23} & -J_{21} + J_{23} & J_{21} - J_{22} \\ -J_{12} + J_{13} & J_{11} - J_{13} & -J_{11} + J_{12} \\ J_{12}J_{23} - J_{13}J_{22} & -J_{11}J_{23} + J_{13}J_{21} & J_{11}J_{22} - J_{12}J_{21} \end{bmatrix}^T \tag{5.11}$$

Hence, converting from end-effector Cartesian velocity to joint velocity will be:

$$\begin{bmatrix} \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_5 \end{bmatrix} = \underline{J}^{-1} \begin{bmatrix} \dot{x} \\ \dot{z} \\ \dot{\theta} \end{bmatrix} \tag{5.12}$$

After the transformation from the Cartesian velocities to joint velocities these values will be integrated in order to obtain the desired values of the joint angles as shown in Fig. 5.4.

5.1.1.3 Process algorithms

Fig. 5.5 presents an overview scheme of the functions in MATLAB and WinDDC software and how both programs are together connected. Image processing algorithms is coded in C language and executed in 150ms using m-files in MATLAB program, this means we get every second about 7 updated pictures (7 frames per second – 7fps) of the scene. The image processing in RGB format is performed on sub-window of 500 X 20 pixels. The proposed code consists of main program and two subroutines. The first will calibrate the color of the marker. The second subroutine will convert RGB image to binary image depending on the calibrated color, after that it will detect the marker and finally it will update the color of the marker.

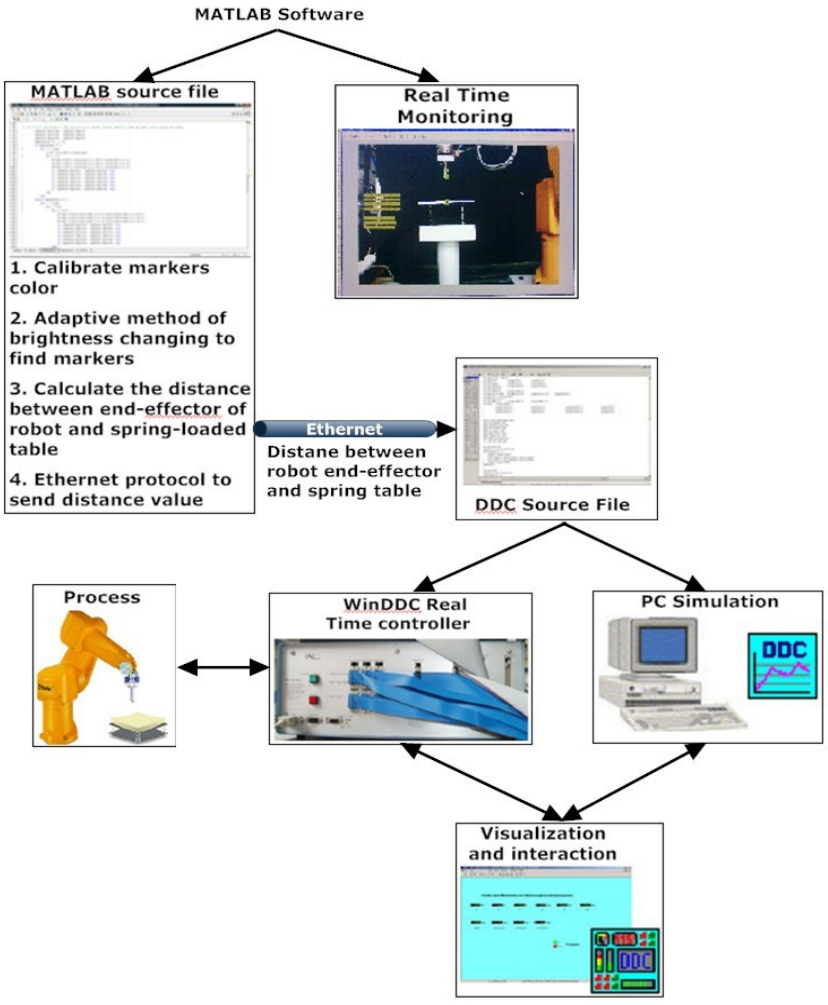

Fig. 5.5 Overview of software equipments

Firstly, the system needs to calibrate markers color which is done in the first subroutine. The green color has chosen as default color for the marker. Fig. 5.6 illustrates the algorithm of color calibration which is performed offline. The user will clicks on the target object with the left-mouse button, after that the system will save the values of color components (red, green, blue). When the user clicks right-mouse button, the program will end the loop of pixel-samples

collection. In the next step, the program will calculate the minimum and maximum values of the color components and it will save them as ranges of the target color.

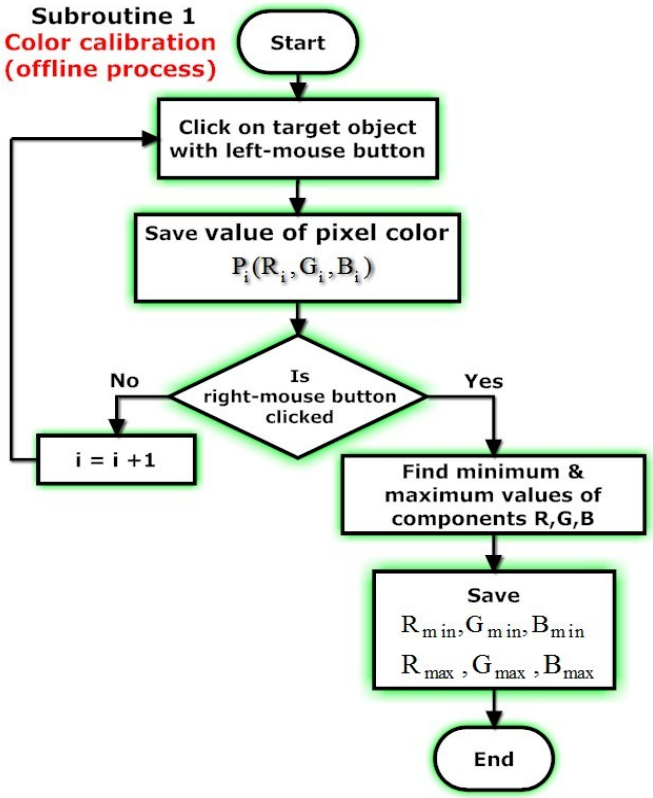

Fig. 5.6 Color calibration algorithm

In the second subroutine, the program detects the three green markers by converting the RGB image to binary image. The values of the binary pixels will be 1 (white pixel), if the corresponded RGB pixels have color values between the ranges of the calibrated color. Otherwise, the binary pixel will be black. In other words, the pixels of the target color will be converted to white pixels in the binary images and the rest will converted to black pixels.

As is well know, every pixel has three values of color intensity (R, G, B) and every value ranges from 0 to 255. These intensity values are very sensitive to the brightness. To avoid this problem we designed an adaptive detection algorithm. In every frame the real color values (R, G, B) of the three markers are compared with markers' colors of the previous frame. If they are different, the markers' colors of the new frame will be saved as a new color data. This means when the surrounding circumstances of the environment change, such as light intensity, vision angle, shadow of the object or anything else, it will not affect the performance of detection algorithm.

Fig. 5.7 presents the second subroutine which consists of two phases: In the first phase the program will use some morphological operation such as "dilate" and "fill" in order to label the white objects in the binary image. In the second phase the program will update the values of

detected markers in order to ensure good detection for the markers in the next frame, if the illumination conditions are changed.

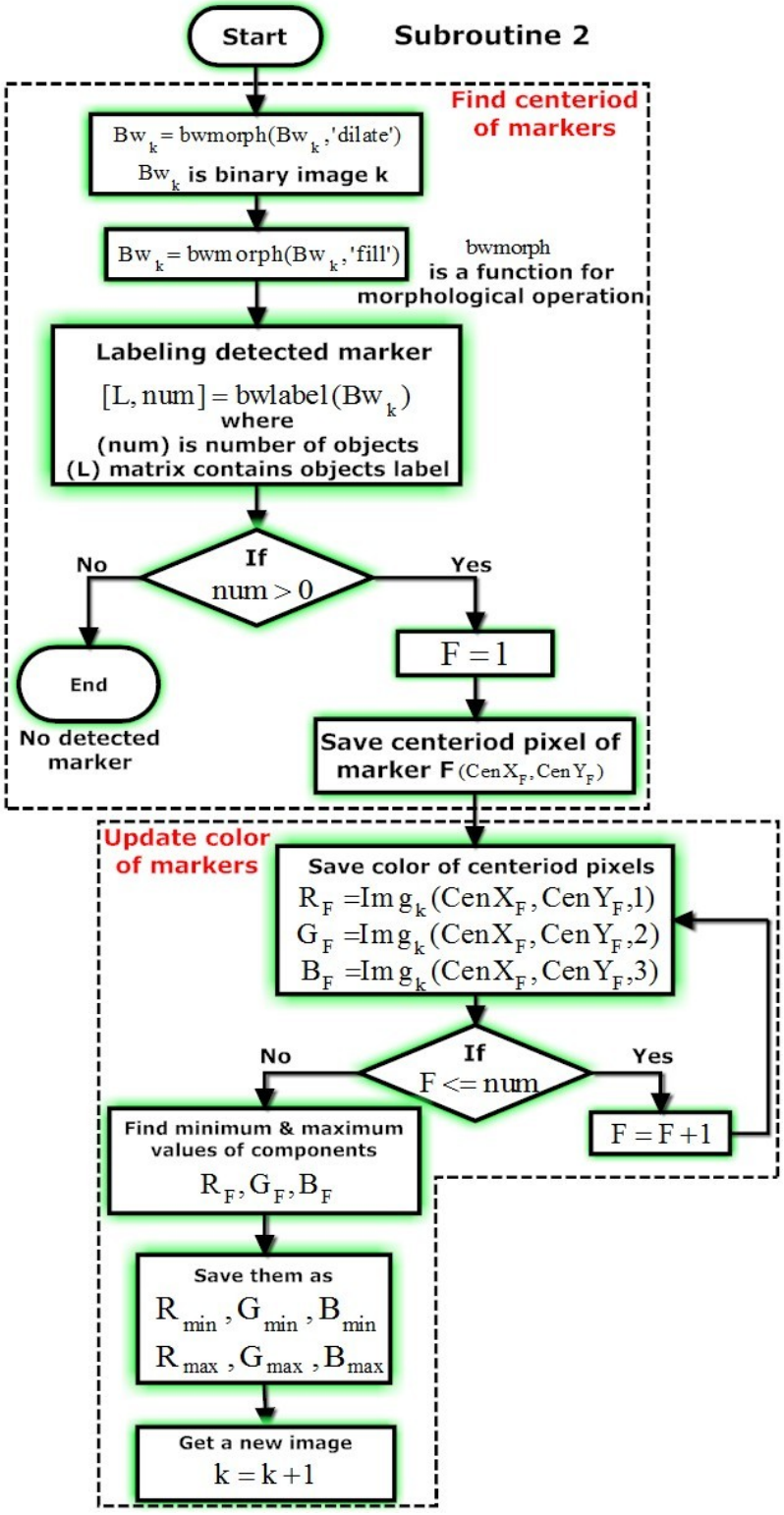

Fig. 5.7 Algorithm of detection and updating of markers color

After calculating the centriod of green markers (see Fig. 5.8), the program will calculate the distance between the centriod of the second and third marker in order to define the distance between the end-effector of robot and spring-loaded table. The calculated value will be sent to the second PC (WinDDC software) by Ethernet TCP/IP using the third subroutine.

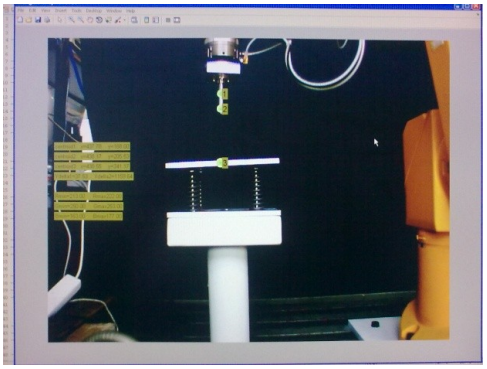

Fig. 5.8 markers detection

5.1.1.4 Modeling and control

This section will discuss the model of the system and the impact control for the case where the manipulator is in free space (no contact with environment) and in contact space.

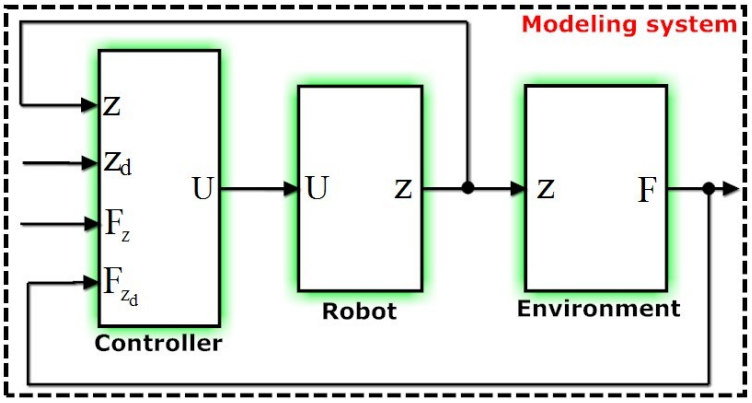

Fig. 5.9 General model of the proposed system

Fig. 5.9 shows the model of the system which consists of three parts controller, robot, environment. The Controller is combining of shared and traded vision/force controller, this calibrates two PID controllers, one for position and the second for force as shown in Fig. 5.10. This choice is justified by its simplicity, easy of use and good results. We have used only proportional term without integral and derivative terms because integral term accelerates the movement of process too much and it can cause big value of overshoot which can do harm to environment or robot and derivative term is highly sensitive to noise in the error term, and can cause a process to become unstable. Furthermore, there is no need for a more advanced controller if the results of the P controller are satisfactory. Three limiting elements are used to ensure that the speed of robot will not exceed maximum values when the error is too large.

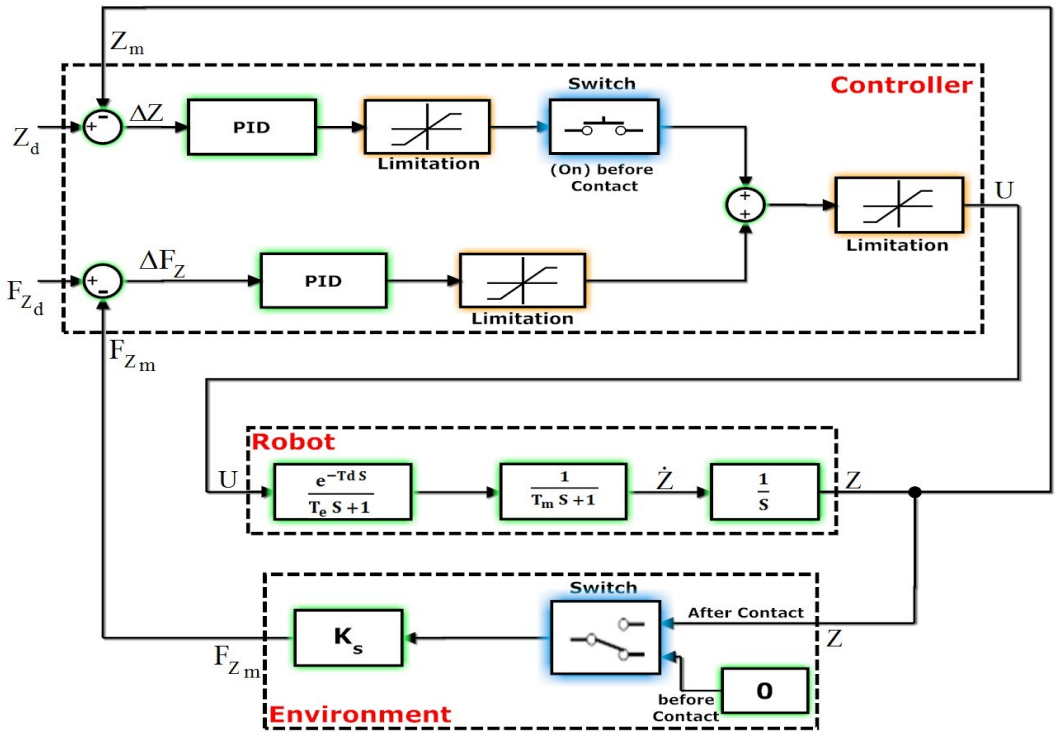

Fig. 5.10 MATLAB model of the proposed system

Robot dynamics is actually modeled for one direction, for the Z axis, there are three parameters in this model: dead time T_d = 0.001 s, this delay occurs because of the control system. Electrical equipment causes leg in the system, which is measured by comparing the desired current with actual current using oscilloscope. This is electrical time constant, T_e= 0.0005 s. The mechanical equipment also causes leg in the system, which is measured from step response of velocity to step of motor current (voltage). This is mechanical time constant, T_m = 0.15s

$$G_1(s) = \frac{Z(s)}{U(s)} = \frac{1}{0.15\,s + 1} \cdot \frac{e^{-0.001\,S}}{0.0005\,s + 1} \cdot \frac{1}{s} \tag{5.13}$$

$$G_1(s) = \frac{e^{-0.001\,S}}{75.10^{-6}\,s^3 + 0.1505\,s^2 + s} \tag{5.14}$$

The environment is modeled as stiffness, as follows:

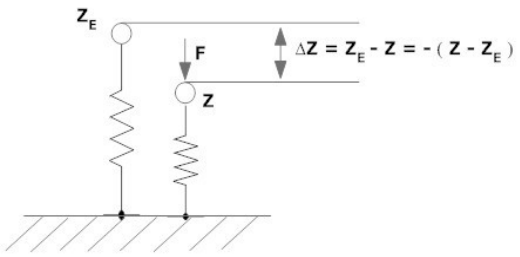

Fig. 5.11 Relation between position and force

$$\vec{F} = -K_s(Z - Z_E) \tag{5.15}$$

Here K_s is the stiffness of the environment, Z_E is the position of the environment at rest and Z is the position of the robot's end-effector. According to the position of the end-effector of robot the controller is divided into two stages: free motion controller and force controller.

Free motion control

Free space motion could be considered as a previous step before starting contact task. During this stage, the robot is approaching the environment at a specified velocity and acceleration which are calculated from position error. By simplifying Fig. 5.10 for free space stage we can come to Fig. 5.12 which shows the controller only in the free space stage. The criteria for the robot during this stage are to have low position error, the force feedback is always zero because there is no contact force in this stage and the desired force will be considered as constant input.

$$Z \rightarrow Z_d\,, \quad e_Z = (Z_d - Z) \rightarrow 0 \tag{5.16}$$

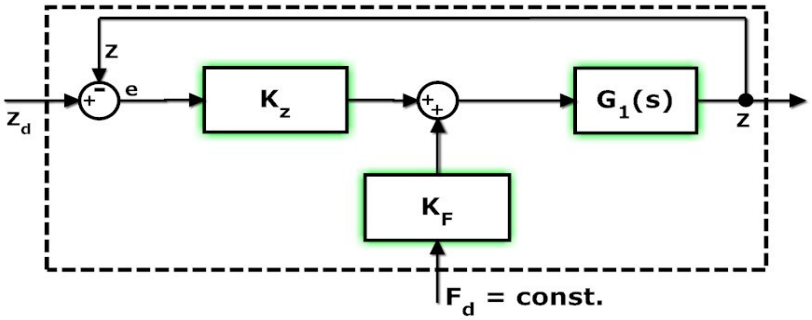

Fig. 5.12 System model of free motion control

According to the distance between the end-effector and the spring-loaded table measured by camera, the speed of the robot will be changed. The closer the robot is to the spring-loaded table, the slower speed is set.

$$Z = G_1(s)\,.\,(K_f\,.\,F_d + K_z\,.\,e) \tag{5.17}$$

Where as K_f proportional force, K_z proportional position

$$e = Z_d - Z \tag{5.18}$$

From (5.17) and (5.18), we can write:

$$Z = G_1(s)\,.\,(K_f\,.\,F_d + K_z\,.\,Z_d - K_z\,.\,Z) \tag{5.19}$$

The transfer function of system in free-space motion is

$$Z = \frac{K_f\,.\,G_1(s)}{1 + K_z\,.\,G_1(s)}\,F_d + \frac{K_z\,.\,G_1(s)}{1 + K_z\,.\,G_1(s)}\,Z_d \tag{5.20}$$

By assuming $T_d \approx 0$ and $T_e \approx 0$ we will get from (5.13)

$$G_1(s) = \frac{1}{0.15\,s + 1}\,.\frac{1}{s} \tag{5.21}$$

Then,

$$Z = \frac{K_f}{0.15\,s^2 + s + K_z} F_d + \frac{K_z}{0.15\,s^2 + s + K_z} Z_d \qquad (5.22)$$

By analysis this transfer function using MATLAB and assuming $K_f = 0.0003$ and $K_z = 0.3$, we will have Fig. 5.13. The manipulator needs about 4 sec to move by 8 cm (0 position) and then control method will be changed to force control

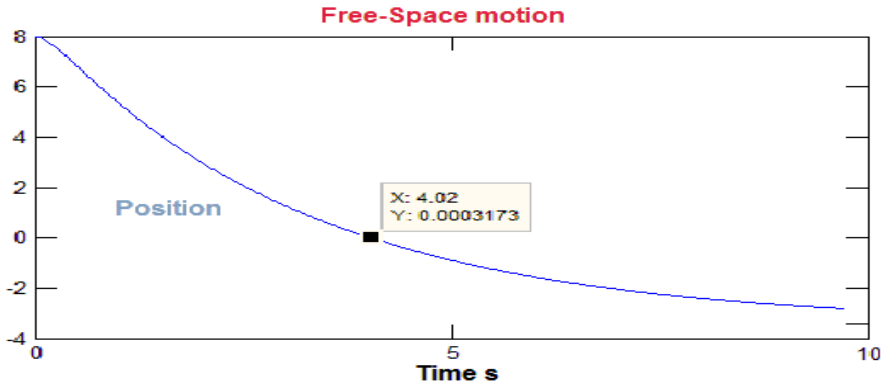

Fig. 5.13 Free space motion

Force control

Force controller must be properly formulated and tuned in order to maintain stability. This can be difficult, particularly during initial contact between stiff surfaces. An effective impact strategy is presented based on a proportional gain explicit force controller (Volpe & Khosla, 1993). The control goal of this method is to regulate the measured contact force (f) to a constant desired force (f_d) along the constraint surface, that is,

$$f \rightarrow f_d \;;\; e_f = (f - f_d) \rightarrow 0 \qquad (5.23)$$

Two types of explicit force control – direct explicit force control (force based) and indirect explicit force control (position based) have been developed (Volpe & Khosla, 1993). In our work force based explicit force control is used which computes the force error by comparing between the reference and measured force signals, processing them, and providing an actuation signal directly to the plant. By simplifying Fig. 5.10 in contact space we can come to case Fig. 5.14 which shows the controller only in the contact space stage.

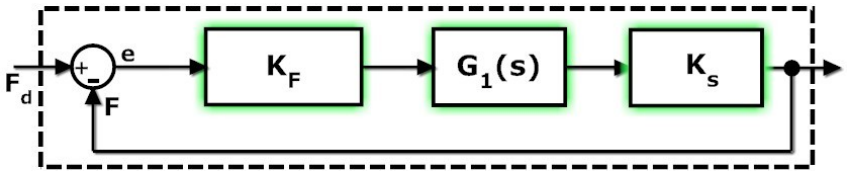

Fig. 5.14 Contact stage control

In this stage the vision feedback will be disconnected and the open loop transfer function of this stage:

$$\frac{F}{F_d} = K_f . K_s . G_1(s) \quad (5.24)$$

Whereas k_s is the stiffness of the spring loaded table, in the close loop system we get:

$$G(s) = \frac{K_f . K_s . G_1(s)}{1 + K_f . K_s . G_1(s)} \quad (5.25)$$

From (5.21) and (5.25):

$$G(s) = \frac{6.667 . K_f . K_s}{s^2 + 6.667s + 6.667 . K_f . K_s} \quad (5.26)$$

By comparing denominator with standard second order system we will get the relation for damping coefficient and resonant frequency parameters and proportional force controller k_f which shown in (5.27) and (5.28)

$$\xi = \sqrt{\left(\frac{1.498}{K_f . K_s}\right)} \quad (5.27)$$

$$\omega_0 = \sqrt{6.667\, K_f . K_s} \quad (5.28)$$

Our experiments were performed with two different spring-loaded tables with different stiffness, the first one was medium stiffness environment and the second hard stiffness environment. Then maximum overshoot (maximum peak) and peak time were calculated according to (5.29) and (5.30):

$$Z_p = e^{-\frac{\xi\pi}{\sqrt{(1-\xi^2)}}} \quad (5.29)$$

$$t_p = \frac{\pi}{\omega_0 \sqrt{(1-\xi^2)}} \quad (5.30)$$

5.1.1.5 Simulation and experimental results

This section will present the simulation results and it will compare them with the experimental results with and without vision feedback in two cases: In the first case control is applied on the medium stiffness environment (spring-loaded table with $K_s = 8$ N/mm) and in the second case it is applied on hard stiffness spring-loaded table ($K_s = 88$ N/mm).

By analysis the transfer function of the system using MATLAB and according to (Chaing & Shen, 2004) we can determine the values of proportional gains for every case, Fig. 5.15 shows the simulation with Proportional controller for vision and force loops, $K_z = 0.3$, $K_f = 0.003$. Notice that the impact force is very low even though the end-effector of robot is in contact with medium stiffness of the spring-loaded table $K_s = 8$N/mm. By using equations (5.27) and (5.29) to find the maximum peak $Z_p = 0.88\%$, i.e. when $F_d = -30$ N, impact force will be about -0.26 N. Improving the impact force in our experiments has not affected on the system speed compared to the case of controlling without vision system. At the beginning the manipulator moved with fast speed and decelerated by approaching the spring-loaded table.

Notice that in experimental curve the system speed is faster than the speed in simulation and the impact force is a little bit bigger than impact force in simulation. This is because the mass model of manipulator is not included in the simulation model for simplifying the calculations.

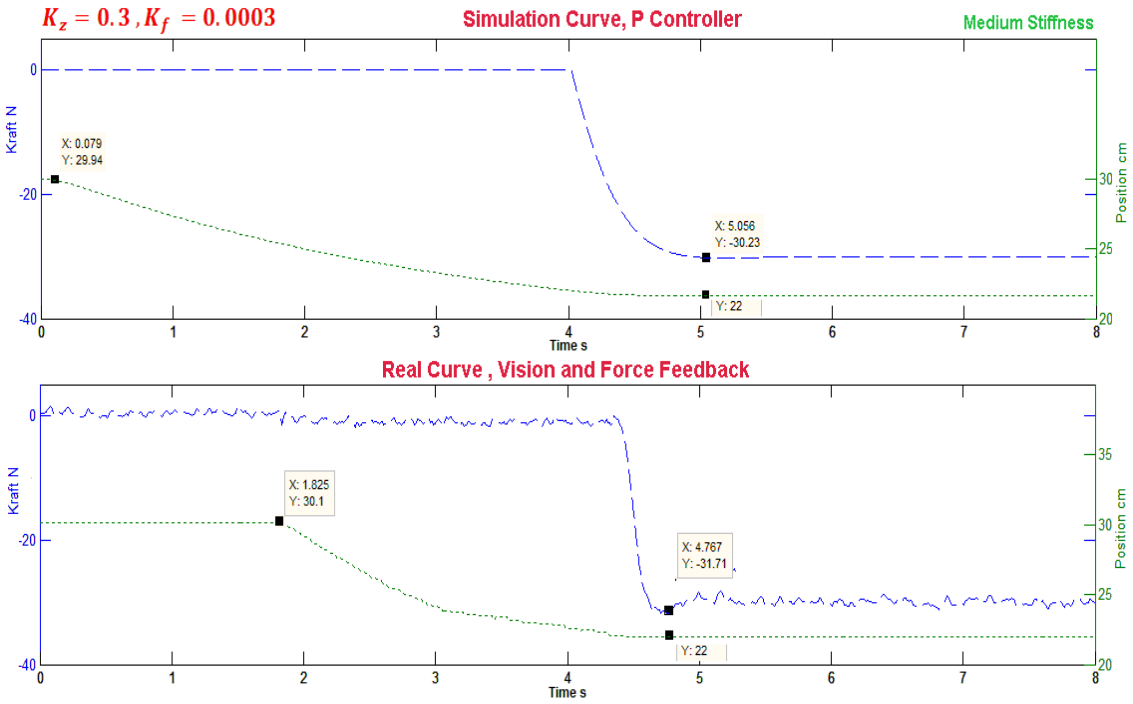

Fig. 5.15 Results of medium stiffness experiment

Fig. 5.16 presents the behavior of the manipulator using only force feedback and medium stiffness spring loaded table. The system needs more than 8 seconds to move 10 cm, and this illustrates the benefits of combination of vision and force feedback.

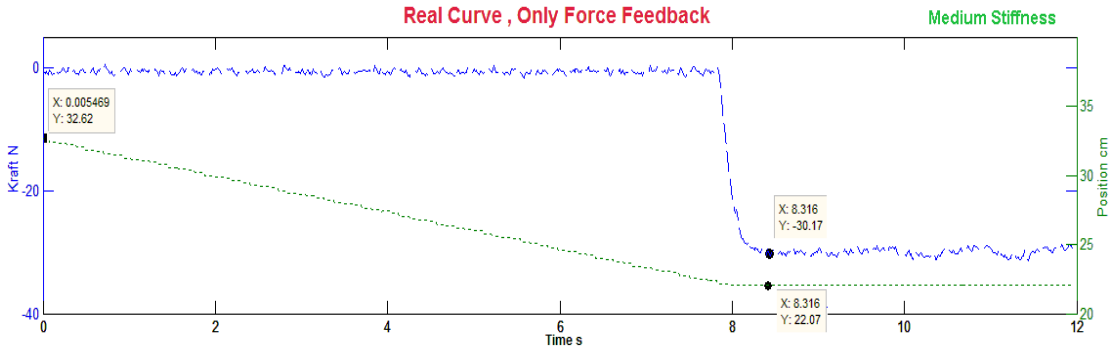

Fig. 5.16 Result using only force control

Fig. 5.17 proves the effectiveness of the technique also with hard stiffness environment, the stiffness of spring-loaded table is about 88 N/mm and the impact force is about 7 N. Notice also the effect of the vision feedback: The manipulator moves only 3 cm in 8 sec in the first diagram of Fig. 5.17. Whereas, in the second diagram of Fig. 5.17 it moves 8 cm in 6 sec.

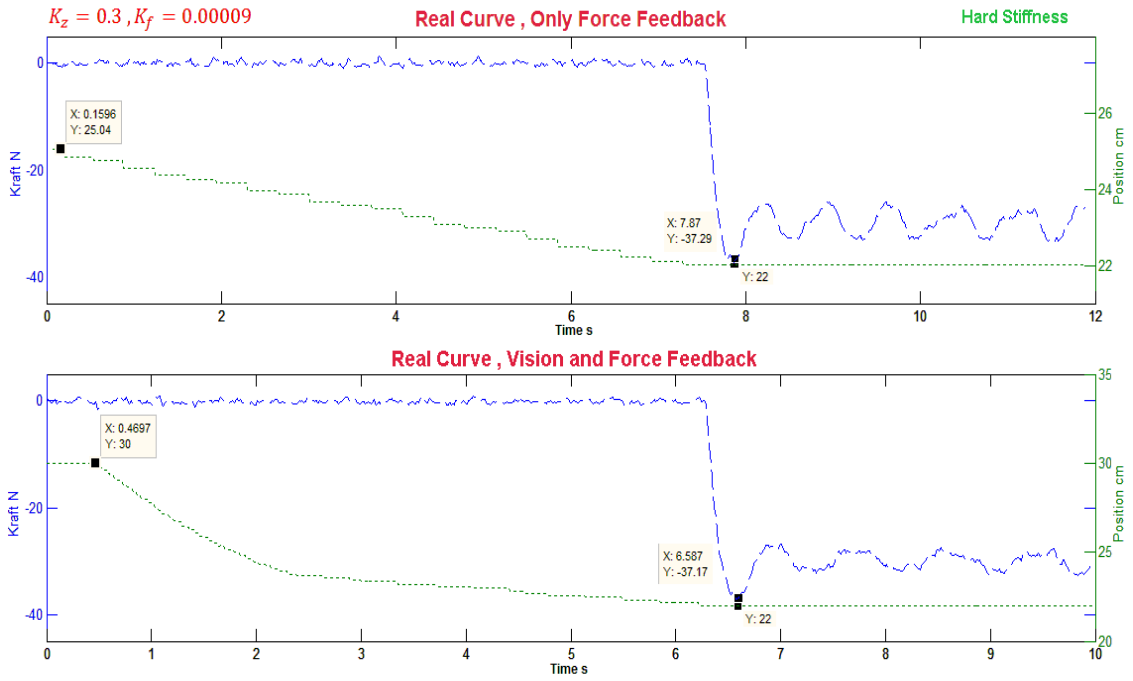

Fig. 5.17 Results of hard stiffness experiment

5.1.1.6 Discussion and conclusion

This section has presented the analysis and experimental testing of vision/force robot control to improve the impact with two kinds of environment's stiffness (medium and hard stiffness). It has shown the advantages of vision feedback in comparison with impact control without visual feedback. All performed experiments concern improving performance of the robot by switching from free space motion to constrained force control. The case presented in this work was relatively simple (one axis because of the available controller of the robot system, Jacobian matrix was calculated only for three joint of the robot). However, it seems that this approach has potential of improvement in more complicated application.

5.2 Vision/force integration for grasping tasks

Recently, vision and force control are operated in a more complex varied environments, which required a high demand for adaption, flexibility and fast performance. More information and features about the environment are required whenever the tasks are more complex and multifarious such as storage and retrieval tasks which are spread everywhere, for example in libraries, factories, warehouses, pharmacies, supermarkets and etc. In this work, it is suggested a robot system to store and retrieve inaccurately placed objects according to their alphabetic/numeric codification system. The vision system will detect the objects' position and orientation and will analyze the alphabetic/numeric coding system using SIFT algorithm. After that the system will describe how it will grasp the objects by combining the vision and force feedback. This section will focus on the library automation as real application for this work.

5.2.1 Library automation with the help of vision/force control

Automation systems in the library are classified into two types: 1. Manual sorting system: Materials are transferred from check-in point to the first store stage automatically, here should be one or more employees sorting the materials according to the types (science, romance, history and etc.) and putting them inside the book trucks. After that other employee moves these book trucks to the floor and department where the book should be shelved. 2. Automated sorting system: It consists of a wide range of conveyors which cover the whole library (also between floors), this system sorts and transfers the materials to their departments. Anyway, these libraries are half-automated because human is still needed in every department to shelve and rearrange the materials by hand. Storage and retrieval tasks in general have the same steps, like searching specified objects, reading and analyzing their codes, grasping them and rearranging them according to their alphabetic codification system in the shelves. This kind of tasks are very boring for the human, and they need a lot of time, a lot of effort and people to do them. Not only that but also in some places like libraries and pharmacies where the workers are skilled, a large part of the normal workday of such staff members is taken up with reading the code of objects, searching about their shelves and rearranging them. This work presents briefly the problems of the previous work and suggests a new robotic system with the help of vision system, image processing algorithms and force feedback in order to solve the main problems of this task. This work is concerned especially with libraries scenario because of many reasons: 1. This scenario is the practical application for the storage and retrieval. 2. According to our knowledge, until now there is probably no full automated library. 3. If the problems of automating a library are solved, the results could be used in a lot of other applications by some changes in details, like in warehouses, in factories, in pharmacies, etc

By inspecting the previous works in this field (see Chapter 1), It can summarized that: two technologies have been used in the most developed automated library: 1. Radio frequency identification (Dhanalakshmi & Mamatha, 2009), which helps to automate processes and allows identification of large number of tagged objects using radio wave. 2. Automated book

storage system: Many inventions which have to do with library automation were concerned with removing the requested books from their prescribed locations in the bookracks and returning them using sorting system and conveyors (Yoshie, 1997) or with the help of robotic systems (Timothy & D.Plutt, 2003), (Nakano, Y.Kihara, Sakimoto, & Hayashi, 2003) and (Suthakorn, Lee, Zhou, Thomas, & Choudhury, 2002). The system in (Yoshie, 1997) has some disadvantages, such as: In every requested or filed order the system transports one complete container even if the customer needs only one book. This means that the system will bear additional encumbrances which can be dismissed, such as waste of energy and more maintenance. The codes of all stored books in the container should be read when the container is brought into the stack room, in order to update the location of the books even if only the pose of one book in the container is changed. This means waste of time and in some cases the location of the book will not be known until the container is brought back to the stack room. On the other hand, other works previously mentioned have proposed different designs and mechanisms of robotic system to automate the library. The disadvantages of these works summarized as follows:

1. Many additional hardware equipments in previous systems could be eliminated by using visual system.
2. All locations of the shelf section storage in the library, such as bookracks and books locations should be prescribed in detail, which requires huge capacities of memory, so if any change in library structure happens, the system should be every time updated.
3. If one book is placed in wrong location, for some reason, there is no possibility to find this book again using the previous systems.

So using vision system in this kind of tasks will save costs, time of fault detection and maintenance operations, it will make the system more adaptive to handle different types of workplaces and it will improve the performance of the system without needing previous information about the environment.

Another two works have recognized the importance of using vision information in the proposed task (Tomizawa, Ohya, & Yuta, 2003) and (Heyer, Enjarini, Fragkopoulos, & Graeser, 2012). However, the proposed algorithms of extraction and return the book from the bookshelf are still simply addressed with many limitations, e.g. robot will grasp only the most right book or an enough space should be kept between the neighbor books, etc. Whereas, In (Prats, Sanz, & Pobil, 2005) the authors have had an opinion similar ours that combining vision and force control is indispensable in such tasks. However, this work has used two-finger gripper and it has assumed that the books are not pressed together on the shelf in such a way as to impede the insertion of the gripper fingers. Furthermore, the hybrid vision/force control has implemented in a simple way in only two directions.

In our work, the proposed vision system detects objects' position/orientation, characterizes and classifies the objects, identify the codes assigned to objects (SIFT features). Furthermore, the visual information will not only be used as a simple feedback or as desired position estimator but it will also extract the relations between objects to define the most appropriate

control mode for every task, either by switching between different types of control modes or by determining the values in selection matrix S. This work has used all possible types of vision/force control combinations and it could use different structures of the control in different subspaces in the same task to ensure the successful of grasping. Our proposed control system is inspired from human performance during extracting a book from a shelf. Where, in some cases the human uses only two fingers to grasp the book, whereas in other cases he/she uses three fingers to extract the book, especially when the books are stuck together and there are no possibility to enter the index finger and the thumb around the target book. Using our automatic decision system (see Chapter 4), the robot system will be able to decide; which directions should be force controlled and which directions should be vision controlled, how the robot will grasp the target book using two fingers or three fingers. Hence, in the proposed algorithms, combining vision and force information will eliminate the drawbacks of the previous works represented by complicated transport mechanisms such as waste of energy, more maintenance, etc. In addition to that, using the proposed automatic decision system will reduce the limitations of the previous works such as existence of adequate space between the neighbor books. The proposed system can handle different situations, e.g. when book has no neighbor books, when it is sloping or even stuck between two other books.

5.2.1.1 Experimental equipments

This section describes the experimental setup used to show the feasibility of using vision and force in complicated tasks.

Hardware

Fig. 5.18 presents the overall experimental setup which consists of:

- ① Stäubli RX90 robot.
- ② JR3 (120M50A) is a six component force/torque sensor and the effective measurement range which used in this application is ± 200 N for force and ± 20 N.m for torque.
- ③ Sony XCD-700 is a firewire CCD camera with 1024 x 768 pixel resolution, easy to install and ideal for a variety of applications such as machine vision microscopy and factory automation. Its frame rate is 15 frames/s and the transfer rate is 400 Mbps with IEEE 1394 digital interface.

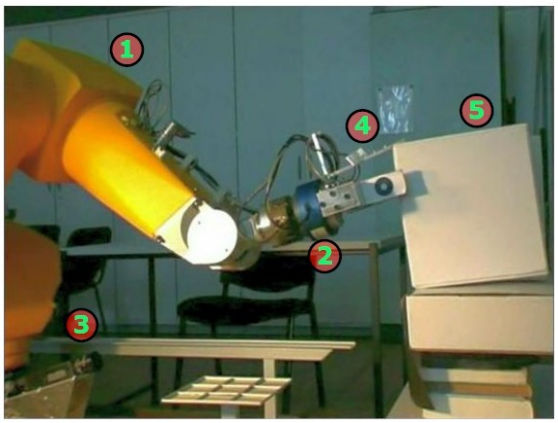

Fig. 5.18 Experimental setup

- ④ The end-effector is installed on the collision protection device. In our application the end-effector is designed as 3 fingers end-effector. Two parallel fingers which hold

the object have digital input (0 open /1 closed) and the third finger which gets the object out is mounted above them (see Fig. 5.19)

- ⑤ Environment which consists of bookrack with different books on it.

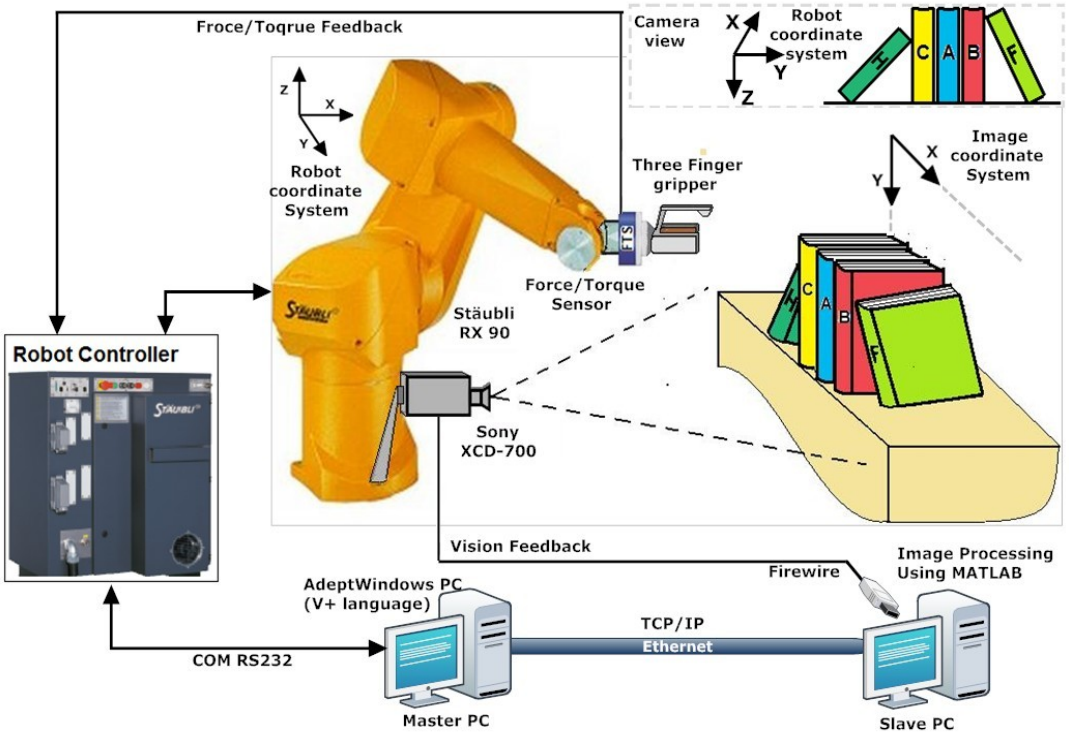

Fig. 5.19 Overview of hardware equipments

Fig. 5.19 demonstrates the structure of the robot system and environment components. The environment is actually bookrack or shelf with few different books on it, which have different position, orientation, characteristics and code. Three fingers end-effector is installed, and the Sony XCD-700 is mounted on the base of the robot. The goal is to detect the objects, define their characteristics (position, orientation, length, width and etc), identify the code for each of them, determine which is the first object to be grasped and how it will be grasped (with or without force), and transporting the objects to new bookrack or new place, in a way, that all the objects will be rearranged according to the alphabetic/numeric code system.

Software

The vision program is coded in C language and executed using m-files in MATLAB software. Robot control is coded in Adept V+ language. In addition to being the complete programming language, V+ is also a complete operating system that controls equipment connected to Adept controllers. The MATLAB program and Adept V+ are connected using TCP/IP protocol.

As shown in the Fig. 5.20, MATLAB program receives from Adept V+ the following: 1. The actual position of the robot 2. The actual force of the force sensor 3. The status information about the current robot motion (if the robot still moves or the motion is finished). MATLAB

program sends to Adept V+ the desired values of robot position and speed. In addition to that, MATLAB program and Adept V+ are fully integrated, which means that all instructions in the V+ language can be sent and used by MATLAB. The system needs at the beginning to calibrate the camera which is mounted in the base. This is done, by taking four points in the workspace, position of which is known in both coordinate systems (robot coordinate system and camera coordinate system). Then the transformation matrix will be calculated in MATLAB program for a two-dimensional projective transformation. After every object has been transported, the system will update the image and repeat the image processing to test if something unexpected happened.

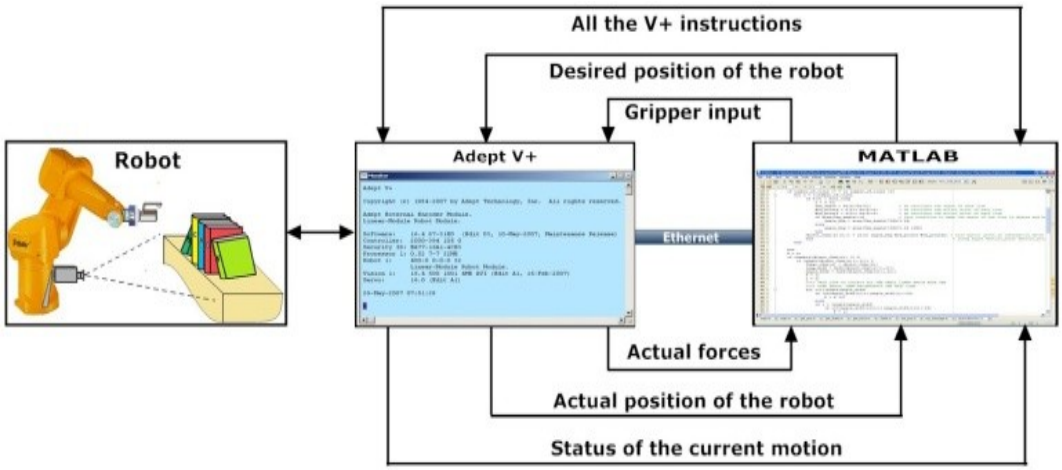

Fig. 5.20 Software of the system

5.2.1.2 Process algorithms

As shown in the Fig. 5.21, the steps of the process algorithms are the following: After receiving the image from vision it will be filtered by Canny filter and processed by morphological operations. Then the system will detect all edges in the image, calculate the junctions (intersection-points)/ endpoints and segment the line to detect all connected lines which form objects in the image. Next steps characterize and classify objects such as defining the length, width, corners, angles, shape etc. for every object and comparing them with target objects characteristics. After detecting the target objects, the algorithm will identify every code of every target object using SIFT feature. All the vision algorithms are explained in details in Chapter 2.

In the next step the system will determine the gripping order algorithm of objects depending on position priorities and code priorities. After that the system will analyze the relation of poses between the first target object and the surrounding objects, in order to define the grasping algorithm (if the task needs the force feedback or not). If the force feedback is needed the system will integrate the visual servoing and force control together to shelve and to rearrange all the target objects according to their codification system.

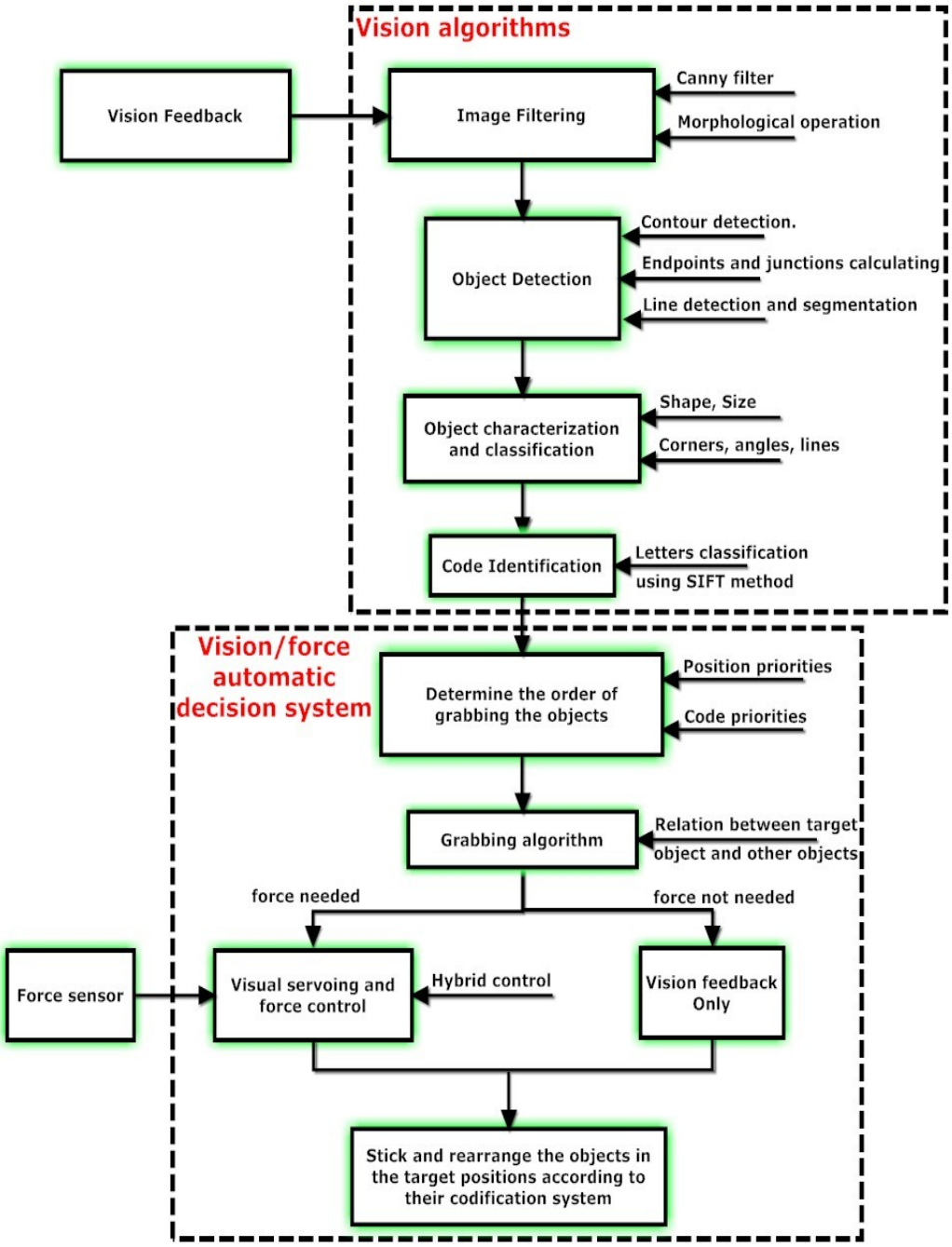

Fig. 5.21 Process algorithms of the automated sorting system

5.2.1.3 Grasping order algorithm of objects

After detecting target objects, their characteristics (position, orientation, length, width etc.) and reading the code of each object, the system will determine grasping order of objects. In other words, what is the first object that should be grasped? Answering this question depends on two priorities, position priority and code priority.

Position priority

Here the system will review the position and orientation of each object. The first object, which should be grasped, is that one which locates above the other objects. It is impossible to grasp them before grasping it. Fig. 5.22 and equation (5.31) show the height level of the object.

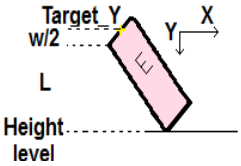

Fig. 5.22 Height-level

$$Height_{lev.} = Obj_{pntY} + Obj_{leng} \cdot abs(\sin(Obj_{ang})) - Obj_{wid} \cdot \frac{abs(\cos(Obj_{ang}))}{2} \qquad (5.31)$$

If two objects have the same height level, the object which should be grasped firstly is the object which has the least stable position (i.e. its angle is away from stability angle 90 degrees), and there is risk of falling would the robot try to grasp the other.

Code priority

If the objects have the same height level and the same stability, the system will compare between the codes of the objects, the first object will be grasped with priority according to alphabet arrangement. As shown in the Fig. 5.23, the first object which should be grasped is B, because it locates above the others. Then the objects H,G,A and E have the same height level, so grasping order will depend on the stability, and it is clear that H is the less stable than the others. Because of that as the next object H will be grasped and then E. After that, the objects which remain are A and G, they have the same level of height and the same level of the stability, so the grasping order this times will depend on the alphabetic/numeric arrangement. The robot will grasp A and then at the end G.

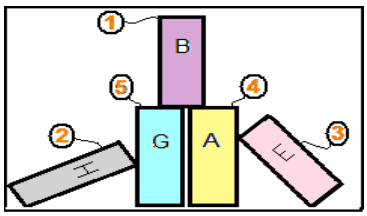

Fig. 5.23 Different poses of objects

5.2.1.4 Control algorithms

As is known vision and force control are not very new topics, and some of robot applications try to combine visual and force information. The challenge here is not how to get information about the environment but how to analyze and understand this information, how to describe the environment in real time, and then how to choose what is the most appropriate control mode for specific tasks. In this work, the visual information will not only estimate the desired position but it will also extract the relations between objects to define the most appropriate control mode for every task, either by switching between different types of control modes or by determining the values of selection matrix S. The force control will be necessary and helpful in many cases, e.g. when the target object is placed between two other objects and there is no sufficient gap to enter the parallel fingers of the end-effector between them. In this case the position control alone will not be enough to perform the task, so the third finger will press on the target object with a specified

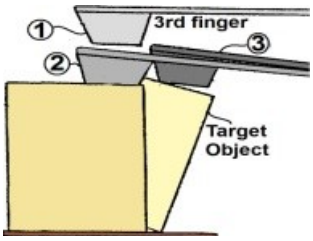

Fig. 5.24 Getting out an object

force/moment and pull the target object out in a way that the two parallel fingers can grip the object as shown Fig. 5.24, like a human trying to get out a book stuck between other books.

In the proposed scenario, the object is placed on a bookrack or a table, the pose (position and orientation) is not accurately known. The first step is to align the robot gripper with the object in a way so that the object can be grasped easily. The alignment between the gripper and the object is specified by the vision feedback as relative position on the camera coordinate system and is then transformed into end-effector coordinate system. Here it is assumed that the x-axis position (see Fig. 5.19) of the objects is constant, so 2D camera is used which gives information about the position in the y and z direction and the rotation around the x-axis in the end-effector coordinate system.

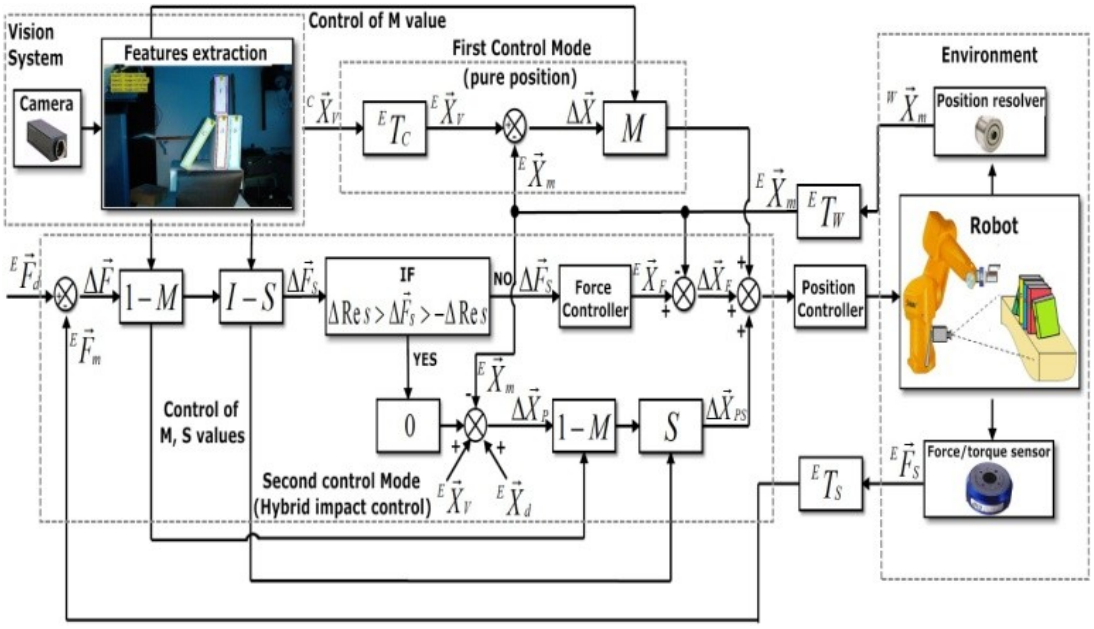

Fig. 5.25 Control algorithms of the system

Fig. 5.25 shows the control structure of the system which consist of: Vision system and two control modes:

1. Vision system: The vision system will send the pose of the first target object to the controller and at the same time it will find the pose of the first target object according to the other neighbor objects in order to define the gripping algorithm, the control mode and the needing of the force control. This is performed by changing the values of the M and S, where S is the selection matrix (see later) and M is binary variable which switches between two control modes.

2. Control modes: The scheme of controller in this work contains different types of control modes. They will be chosen according to the circumstances of the task found by visual information. So it is suggested in this work to use a new variable M which switches between two control modes. More variables could be used in more complicated cases when more than

two different vision/force control modes were needed. If M = 1, the first control mode (pure position control) will be used and the control loop of the second control mode will be disconnected. This means that no force control is needed. With Xor $\vec{X}$ will be further denoted the pose of end-effector in different coordinate systems. With C, W and E the coordinate systems of camera, world and end-effector are denoted. Pose is represented by homogeneous matrix, e.g. ${}^{E}X_m$ or by an equivalent vector:. ${}^{E}\vec{X}_m^T = [{}^{E}x_m, {}^{E}y_m, {}^{E}z_m, {}^{E}\theta_{xm}, {}^{E}\theta_{ym}, {}^{E}\theta_{zm}]$

By simplifying Fig. 5.25 and considering M = 1 we come to Fig. 5.26 which shows the pure position control mode. Vision system extracts the pose of first target object as relative pose in the camera coordinate system.

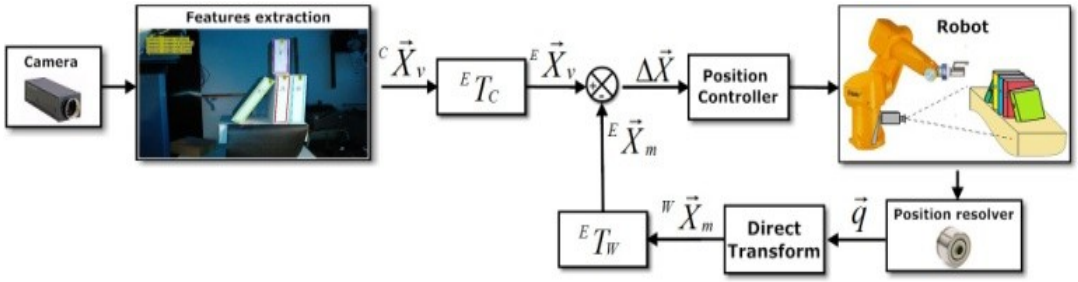

Fig. 5.26 Pure position control mode

The pose error $\Delta\vec{X}$ is calculated as:

$$\Delta\vec{X} = {}^{E}\vec{X}_V - {}^{E}\vec{X}_m \tag{5.32}$$

where ${}^{E}\vec{X}_m$ is measured pose of end-effector, which comes from robot control system, ${}^{E}\vec{X}_v$ is desired pose of end-effector which comes from vision. Poses are transformed from camera and world coordinate system to the end-effector coordinate system using transforms ${}^{E}T_C$ and ${}^{E}T_W$represented in form of 6x1 vectors.

$${}^{E}\vec{X}_m = {}^{E}T_W({}^{W}\vec{X}_m) \qquad {}^{E}\vec{X}_v = {}^{E}T_C({}^{C}\vec{X}_v) \tag{5.33}$$

If M = 0 and by simplifying Fig. 5.25 we come to Fig. 5.27, which shows the hybrid control mode. The desired pose could come from vision feedback ${}^{E}\vec{X}_v$ or from user ${}^{E}\vec{X}_d$ and then the measured pose ${}^{E}\vec{X}_m$ is subtracted from them to calculate the error pose $\Delta\vec{X}_P$:

$$\Delta\vec{X}_P = {}^{E}\vec{X}_v - {}^{E}\vec{X}_m \quad \text{or} \quad \Delta\vec{X}_P = {}^{E}\vec{X}_d - {}^{E}\vec{X}_m \tag{5.34}$$

With the help of vision it will also be decided (by values of the selection matrix S, $S_i = 1$ or 0) which subspace will be force controlled ($\Delta\vec{F}_S$) and which subspace will be position controlled. The desired pose in subspaces which will be position controlled ($\Delta\vec{X}_{PS}$) is as follows:

$$\Delta\vec{X}_{PS} = S \cdot \Delta\vec{X}_P \tag{5.35}$$

$$S \cdot \Delta \vec{X}_P = \begin{bmatrix} S_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & S_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & S_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & S_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & S_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & S_6 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ \theta_x \\ \theta_y \\ \theta_z \end{bmatrix} = \begin{bmatrix} x_S \\ y_S \\ z_S \\ \theta_{xS} \\ \theta_{yS} \\ \theta_{zS} \end{bmatrix} \quad (5.36)$$

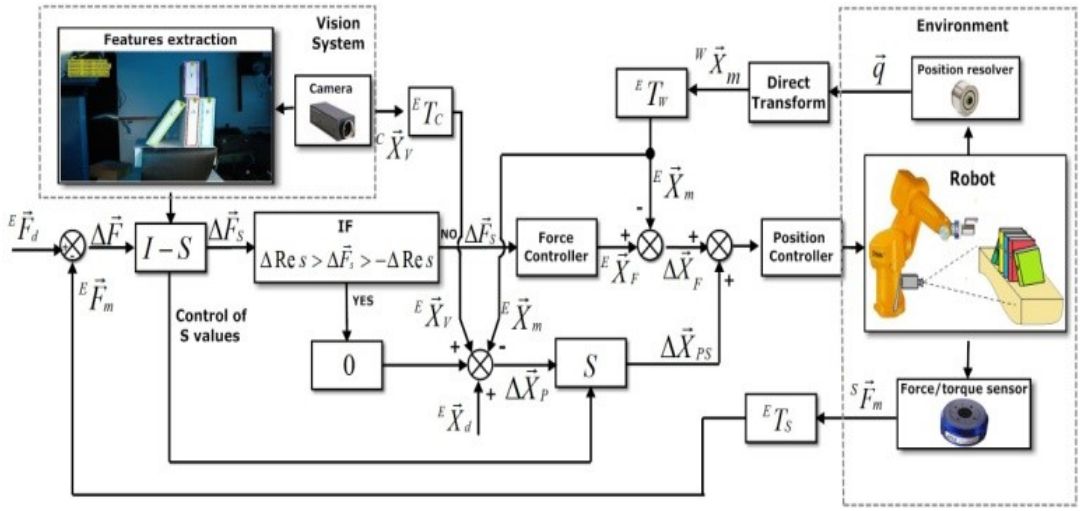

Fig. 5.27 Hybrid control mode

where $\Delta\vec{F}$(6x1) is the force error computed from desired ${}^{E}\vec{F}_d$ and measured force vector ${}^{E}\vec{F}_m$:

$$\Delta\vec{F} = {}^{E}\vec{F}_d - {}^{E}\vec{F}_m \quad (5.37)$$

$$\Delta\vec{F}_S = (I - S) \cdot \Delta\vec{F} \quad (5.38)$$

$$(I - S) \cdot \Delta\vec{F} = \begin{bmatrix} 1-S_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1-S_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1-S_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1-S_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1-S_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1-S_6 \end{bmatrix} \cdot \begin{bmatrix} \Delta F_x \\ \Delta F_y \\ \Delta F_z \\ \Delta M_x \\ \Delta M_y \\ \Delta M_z \end{bmatrix} = \begin{bmatrix} \Delta F_{xs} \\ \Delta F_{ys} \\ \Delta F_{zs} \\ \Delta M_{xs} \\ \Delta M_{ys} \\ \Delta M_{zs} \end{bmatrix} \quad (5.39)$$

Further, components of $\Delta\vec{F}_S$ will be compared with $\pm\Delta Res$, where ΔRes is the threshold of forces/moments. If the force values are not within this range the desired pose ${}^{E}\vec{X}_F$ resulting from force error will be computed by equation:

$${}^{E}\vec{X}_F = A \cdot \Delta\vec{F}_S \quad (5.40)$$

Here A is desired admittance matrix. However, if the force is within the limits, ${}^{E}\vec{X}_F$ will be considered as 0. At the end the sum of all error signals will be calculated by equation (5.41) and it will be sent to position controller which is P controller.

$$\Delta\vec{X} = \Delta\vec{X}_F + \Delta\vec{X}_{PS} \quad (5.41)$$

The position and force control laws in Fig. 5.27 consist of PID action; this choice is justified by its simplicity, ease of use and good results.

5.2.1.5 Experimental results

Fig. 5.28 and Fig. 5.29 present the measured values of pose and force/torque of the robot end-effector to illustrate the grasping tasks of book H and book A (see Fig. 5.19). Both books H and A have totally different poses and therefore distinct processing algorithms. The grasping tasks of them will be solved in different ways.

Fig. 5.28 consists of two diagrams. The first diagram shows the position values of the robot end-effector for x, y and z. The second diagram presents the angle values of the robot end-effector for θ_x, θ_y and θ_z. The vision system will detect the book H, it characteristics and its poses (see Fig. 5.19 book H has diagonal pose). After that vision system will analyze the relation between book H and other books to see if the fingers of the end-effector can enter between the books. In case of book H the system has decided, that the gap between book H and other books allows entering the parallel fingers of the gripper between books. In this case the force control loop is not needed.

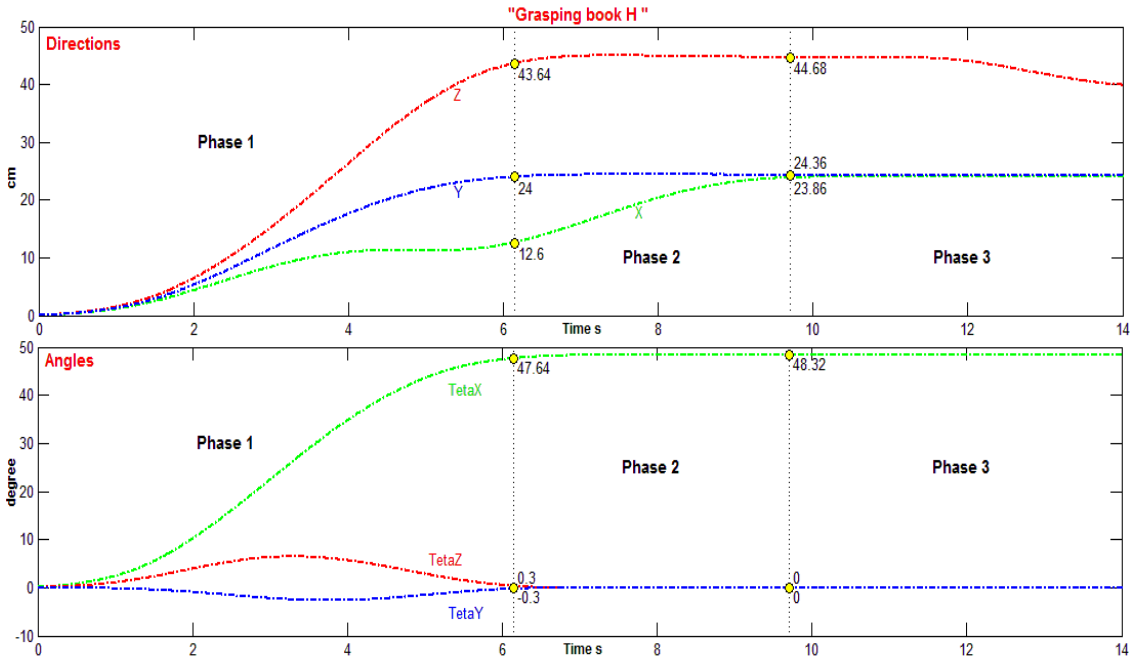

Fig. 5.28 End-effector poses during grasping of book H

The camera in this experiment is 2D, so y, z and θ_x will be vision controlled and x will be position controlled. As shown in Fig. 5.28 the task is divided into three phases. In the first one the robot will align the gripper with the book H in a way that the book can be easily grasped. In the second phase robot will move only in the x direction to position the fingers of gripper around the book H. In the last phase the gripper will grasp the book H and after that the robot can sort or transport it.

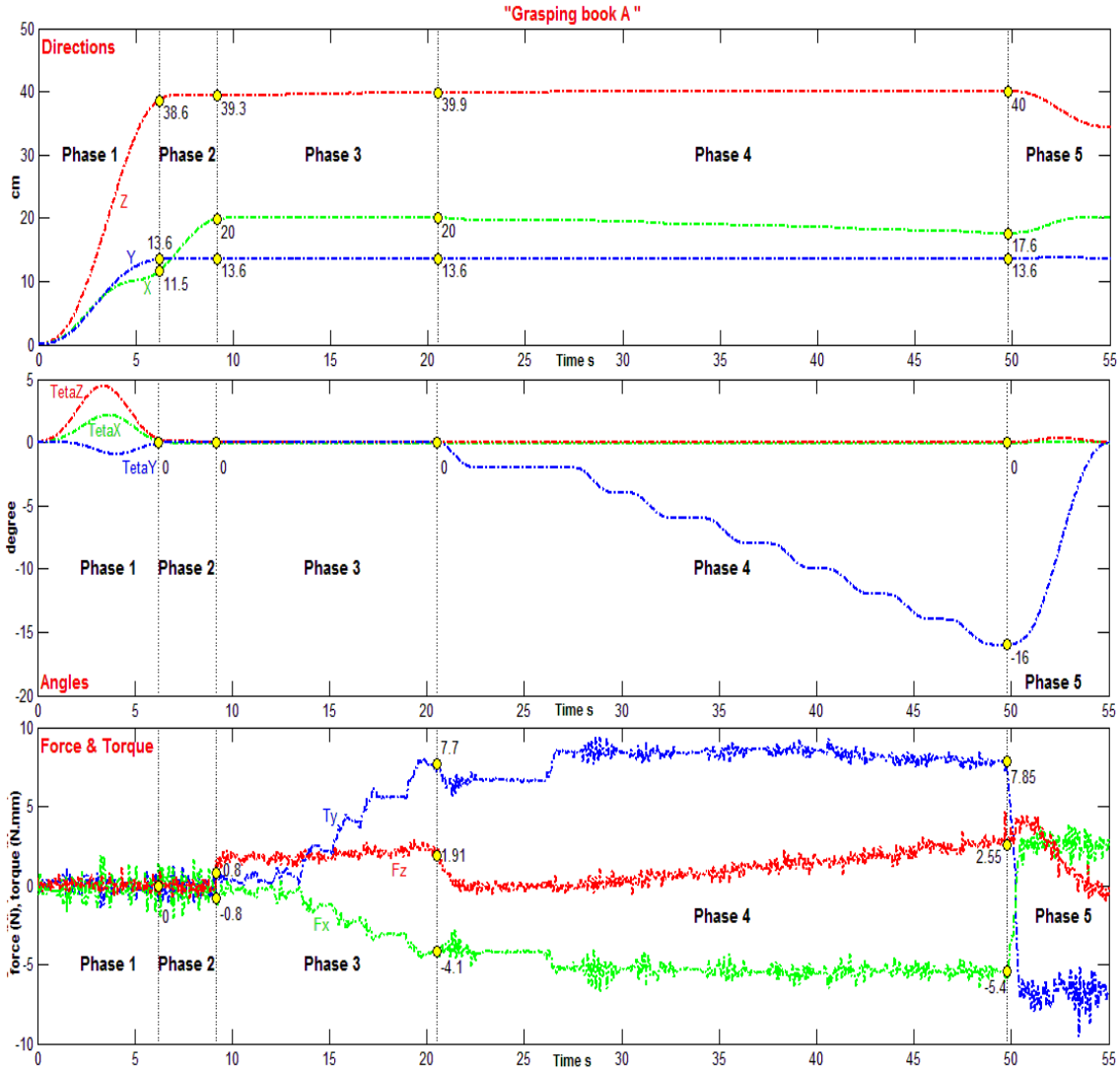

Fig. 5.29 End-effector pose/force during book A grasping

Fig. 5.29 consists of three diagrams. The first and second diagrams present the pose values of the robot end-effector x, y, z, θ_x, θ_y and θ_z. The third diagram shows the force/torque values of the robot end-effector for subspaces F_x, F_z and T_y. As shown in Fig. 5.19 the book A has right angle situation and it is stuck between two other books C and B. After detecting the book A and finding its characteristics, poses and relations with other books, the vision system will recognize that there is not enough gap between the books to enter the parallel fingers of the gripper. Hence, the system has decided to grasp the book in the way which has been explained in Fig. 5.24. As shown in Fig. 5.29, the task is divided into five phases. The first phase in this case is similar to first phase of grasping book H, then in the second phase robot will move only in x direction to position the third finger of the gripper above the book. In the third phase, z direction will be switched from vision control to force control. In other words, in this phase the robot will move in z direction and press on the book until the contact between the robot and the book reaches the desired force F_z (position based explicit force control) as follows:

$$if\ (f_{zm} > f_{zd} + \eta_1)\ or\ (f_{zm} < f_{zd} - \eta_1)\ then$$
$$z_d = z_m - K_1 \cdot sign\ (f_{zd}\ -\ f_{zm}\ ) \tag{5.42}$$

where f_{zd} and f_{zm} are the desired and measured force in z direction, η_1 is limit value of force and K_1 is the gain of the force control loop.

The fourth phase will start, when the contact force reaches the value of the desired force for the first time. The situation in this phase is the following: The robot has good contact with the book (the third finger presses on the book with desired force) and then the robot will try to rotate the book about y axis and at the same time it will pull out the book in x direction slowly while controlling the desired force applied on the book, the same strategy as with human (see Fig. 5.24). As long as the conditions of force control are satisfied, the rotation and pulling out operations will be repeated step by step until the book reaches a pose where the parallel fingers of the gripper can grasp it easily, as follows:

$$if\ \left(T_{yd} + \eta_2 > T_{ym} > T_{yd} - \eta_2\right)\ then$$
$$\theta_{\mathrm{yd}} = \theta_{\mathrm{ym}} -\ s_1\ \&\ x_d = x_m - s_2 \tag{5.43}$$

where T_{yd} and T_{ym} are the desired and measured torque in y direction, η_2 is limit value of torque, s_1 and s_2 are the steps of rotating in y and pulling in x. Equation (5.43) will be repeated until θ_y reaches a desired value in a way that the two parallel fingers can grasp the object. However, if the measured force is outside of the desired force range, robot will further move according to the equation (5.42). In this phase the torque and the angle in the θ_y direction will be controlled at the same time. In the last phase the robot will grip the book A and sort or transport it to another place. On the whole of this task, according to the automatic decision module the subspaces will be controlled as the follows: x and θ_z will be position controlled, y and θ_x will be vision controlled, z will be traded vision/force controlled and θ_y will be shared position/force controlled. As shown in the two tasks, both books have totally different poses and surrounding conditions which make the grasping tasks of them solved in different ways.

5.2.1.6 Conclusion

This section has proposed a robotic system which integrates vision and force information in order to shelve and retrieve imprecisely placed object according to their alphabetic/numeric codification. The proposed system can grasp books even if the target book has no neighbours, slop position or it is stuck between two other neighbours. The system will eliminate the shelving errors that frequently occur when shelving is performed manually, and valuable time spent looking for books that are misplaced when such errors occur.

5.3 Vision/force integration for human robot interaction tasks

Handing-over objects from/to human is an essential step to perform different tasks especially those require physical interaction between the robot and the human. The physical human robot interaction consists usually of two main parts: the human hand and the robot hand. However, in most service robots applications this interaction will not be performed directly between human hand and robot hand, but a transported object will serve as a connection bridge between human hand and robot hand. In other words, the system will transfer objects between the human hand and the robot hand. This scenario could be useful in various applications, e.g. with robot assistants for blind, disabled or elderly people helping them in fetching, carrying things or transporting objects. In other applications the robots serve as members of human robot team as physical support to humans for such applications as space exploration, construction, assembly etc. In every handing-over task there are the giver, the receiver and an object which will be transferred. By handing-over an object from human hand to the robot, the human will be giver and the robot will be receiver. Otherwise, the robot will be giver and the human will be receiver. Our work encompasses both tasks. Here, we will divide the giver or the receiver into three types depending on their behavior during the handing-over:

Fig. 5.30 Human robot interaction

- Positive party (giver or receiver): In this case, this party will play a positive role during handing-over of the objects. In other words, he/she will move toward the other party and tracking his/her hand in order to achieve smoothly handing-over task.
- Neutral party: In this case, this party will try to fix his/her hand in a stable pose, and the other party should move toward them to achieve the handing-over task.
- Negative party: Here, this party will play a negative role; party is e.g. elderly, blind or he/she is doing something else at the same time. In this case, the other party should expect some random motions from it during the task and react accordingly to them.

Table 5.1 presents all the behaviors cases of the parties during handing-over task. If both parties (the receiver and the giver) are negative or neutral parties, the handing-over will not be accomplished. At least one of both parties should perform the task positively by tracking the other party, defining the contact point, searching for contact and also tracking during the interaction phase in order to achieve a smooth handing-over task. The common used approach in the previous works is the case 8, e.g. (Huber, Rickert, Knoll, Brandt, & Glasuer, 2008), (Edsinger & Kemp, 2007) and (Bischoff & Graefe, 2004), where the task is performed exclusively by the human. This means that the robot will bring the gripper into a specified position and orientation and then it will wait until the human places the object between the fingers of the gripper. When the robot detects that an object has been placed in its hand, attempts to grasp the object. In fact, this scenario will not be fit to assistance blind, disabled and elderly people or even to support workers concentrating on their work. In our approach,

we will assume that the human whether a receiver or a giver will be the weakest party (blind, elderly, etc) and the robot will play the main role as a positive party during handing-over task (case 6 or case 9).

	Human	Robot	Handing-over task
1	Negative	Negative	Will be unsuccessfully performed
2	Negative	Neutral	Will be unsuccessfully performed
3	Negative	Positive	Could be successfully performed, if robot is faster than human
4	Neutral	Negative	Will be unsuccessfully performed
5	Neutral	Neutral	Will be unsuccessfully performed
**6	Neutral	Positive	Will be successfully performed
7	Positive	Negative	Could be successfully performed, if human is faster than robot
*8	Positive	Neutral	Will be successfully performed
**9	Positive	Positive	Will be successfully performed (Optimal case)

Table 5.1 Behavoir of parties during handing-over task

In (Cakmak, Srinivasa, Lee, Forlizzi, & Kiesler, 2011) and (Kim, Park, Hwang, & Lee, 2004), the authors have presented different algorithms of grasp planning during handing-over objects between human and robot. These algorithms depend on human preferences, on the kinematic model of the human and on the availability of the 3D model of the object, which means the robot will hand over the object with a configuration planned using a kinematic model of a human and learned from examples given by other humans or with the help of predefined templates of the target object.

Related to that, various papers have focused on analyzing and detecting human body movements especially on hand gesture and facial features, e.g. (Hussain, Siad, Ahamed, Sundaraj, & Hazry, 2011), (Chuang, Chen, Zhao, & Chen, 2011) and (Yaun, Farbiz, Mason, & Yin, 2009). The common properties of them are that they detect the human hand in order to direct or to lead the robot for performing of some tasks without any physical interaction with the robot. In addition to that, they have implemented algorithms which are able to handle with only human hand free of load. Hence, what about if the human hand has carried an object and the task needs the robot to interact with the human physically? Moreover, what about if the user is disabled or elderly?

* The common approach in the previous work

** The proposed approach

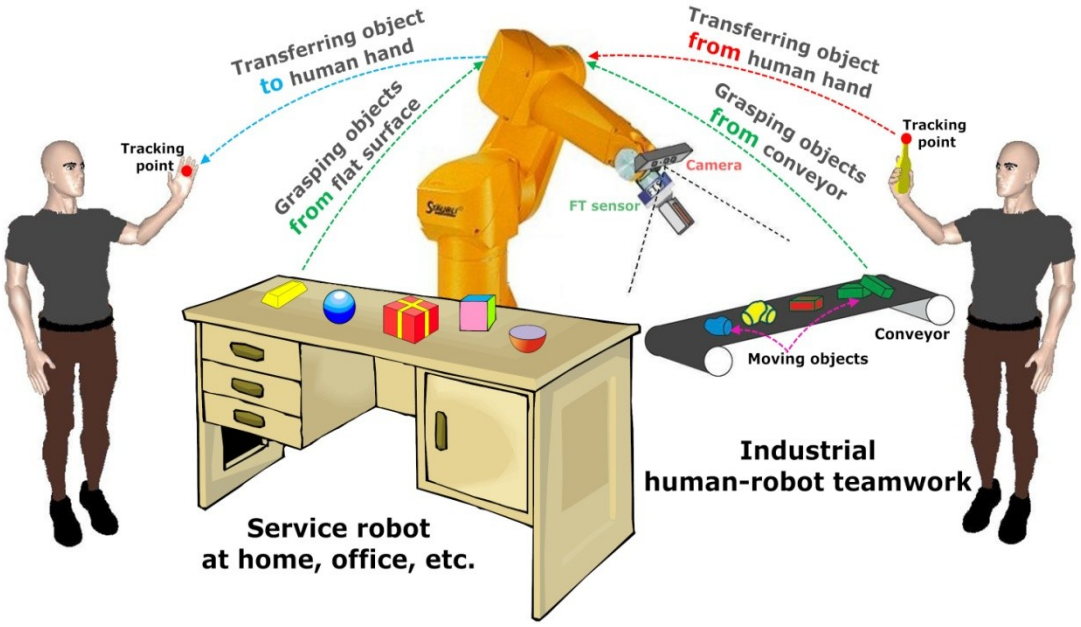

Fig. 5.31 Handing-over from/to human hand

This work will propose an assistance robot system which is able to transfer model-free objects from/to human hand with the help of visual servoing and force control as shown in Fig. 5.31. The proposed robot system is fully automated, i.e. the handing-over task is performed exclusively by the robot and the human will be considered as the weakest party, e.g. elderly, disabled, blind, etc. The proposed system is supported with different real time vision algorithms to detect, to recognize and to track: 1. Any object located on flat surface or conveyor. 2. Any object carried by human hand. 3. The loadfree human hand. Furthermore, the proposed robot system has integrated vision and force feedback in order to:

- Perform the handing-over task successfully starting from the free space motion until the full physical human robot integration.
- Guarantee the safety of the human and react to the motion of the human hand during the handing-over task.

In the proposed system, the robot system should take into account the following problems in order to achieve a successful handing-over:

- How to find out the contact/grasping point of the carried object?
- How to perform the visual tracking of a movable object (carried by human hand)?
- How to ensure the human safety during the tracking phase?
- When can the robot start interacting with the human?
- What should the robot do, when human moves during the physical interaction phase?
- How to confirm that the handing-over is performed (when the robot should grasp or release the object)?

All these procedures lead us to integrate the vision/force robot control as optimal solution for guarantee the safety of the user and for performing the handing-over task successfully.

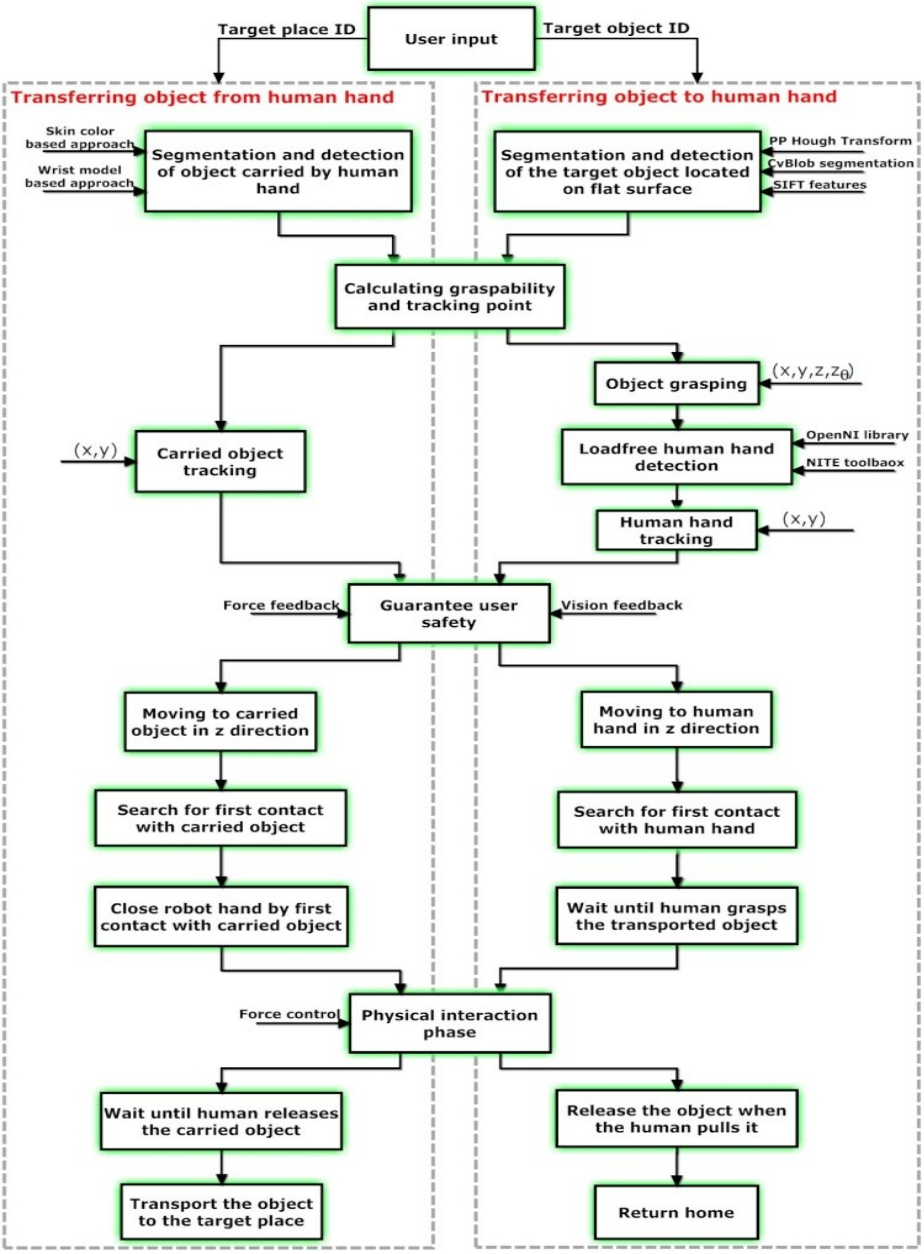

Fig. 5.32 Overview of the proposed system

Fig. 5.32 presents an overview of the proposed system for both tasks: 1. Handing-over from human hand and 2. Handing-over to human hand. Firstly the user will define the type of the task, transferring either from or to human hand. If the task is transferring object to human hand, the user should enter either target object ID or name under which its corresponding SIFT features are saved in the database. In other hand, when the task is transferring from human hand, the user needs only to enter the target place ID (to where the object should be transported, e.g. table, conveyor etc). User interface could be improved in the future to be based on voice commands. The currently proposed system is supported by voice subsystem, i.e. it will announce the current phase (what it is going to do), the status of the operation or

whether some errors have occurred. The proposed speaking subsystem will give the human the opportunity to learn and to understand what the robot is doing now and to be prepared if any error has occurred during the task. It will increase the safety factor between human and robot, especially if the user is disabled or blind.

The next two sections will illustrate the phases of the handing-over task in detail. Firstly, transferring objects from human hand will be presented, whereas the second section will discuss the handing-over from robot hand to human hand. Some of these algorithms are already explained such as calculating graspability and tracking point (see Chapter 4). The main focus of this chapter is oriented toward visual tracking algorithms, safety procedures, physical integration procedures and their experimental results.

5.3.1 Transferring model-free object from human hand

Handing-over objects from human hand to robot hand needs a robot system which is able to decide automatically how grasping task of the objects should be performed, especially when the human will be a negative giver. Fig. 5.33 presents: 1. The receive which consists of robot gripper provided with force/torque sensor and Kinect camera. 2. The giver which is a human hand carrying a model-free object.

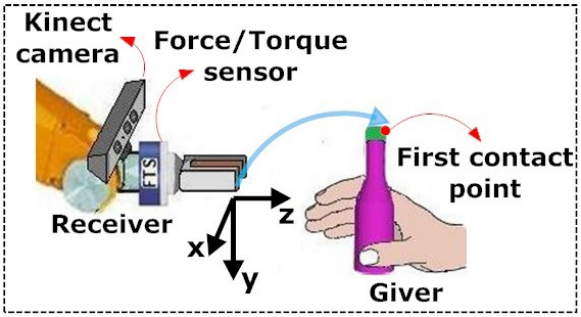

Fig. 5.33 Handing-over from human to robot

This section will focus on grasping unknown objects from human hand. The proposed robot system will perform the following: Detect the hand loaded with the unknown object, track it, distinguish between the human hand and the unknown object, determine the possible contact points where the robot will contact and grasp the object. It will then define the optimal combination of vision and force control in order to guarantee the safety, to ensure the fulfillment of grasping task and to react to the motion of human hand during the physical interaction phase.

5.3.1.1 Experimental equipments

The overall experimental setup, as is shown in Fig. 5.34, consists of: 1. Stäubli RX90 robot 2. End-effector is the Two-finger hand. Two parallel fingers which hold the object have digital input (0 = open, 1 = closed). 3. JR3 (120M50A) multi-axis force/torque sensor and its effective measurement range is ± 100 N for forces F_x, F_y, F_z and ± 10 N.m for torques M_x, M_y, M_z. 4.

Kinect camera is RGBD cameras which delivers depth and color pictures with VGA resolution (640X480 pixels). 5. The target position, where the object should be transported. 6. Model-free object which is carried by human hand. 7. Active human hand. 8. Human will be integrated with the robot system.

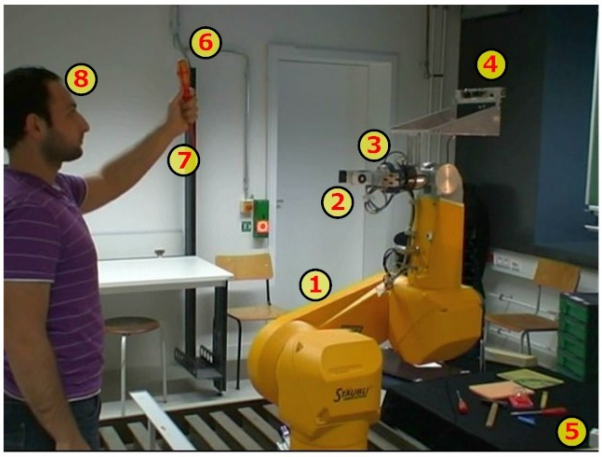

Fig. 5.34 Experimental equipment

Fig. 5.35 demonstrates the structure of the robotic system and the environment components. Two PCs are used. PC1 will control the robot to perform the visual and force tracking tasks with the help of MATLAB program and V+ (programming language of Stäubli robot). A special program is designed to integrate Matlab program with V+ language, which means that all V+ instructions, e.g. read/write pose, read force/torque or robot status, could be written directly in Matlab program. PC2 is connected to the Kinect camera, and all the image processing algorithms are performed in PC2 using C++ language (OpenCV and OpenKinect libraries). PC2 will send the position of the face and of the object as well the status of the task to PC1 in every frame using Ethernet TCP/IP protocol as follow:

$$[face_x, face_y, face_z, Obj_x, Obj_y, Obj_z, Time_{dif}, Vision\_status]$$

Where $(face_x, face_y, face_z)$ is the middle point of the face rectangle (fac_{Reg}), (Obj_x, Obj_y, Obj_z) is the tracking point of the object which will be later the contact point between the object and the robot hand. $Time\_dif$ is time difference between two frames (pervious and current frames). $Vision\_status$ represents the current status of the vision system which could return the following values:

- $Vision\_status = 1$; no face is detected.
- $Vision\_status = 2$; face is detected.
- $Vision\_status = 3$; face and human hand (loadfree hand) detected.
- $Vision\_status = 4$; face, human hand and object are detected (tracking phase could start).
- $vision\_status = 5$; face, human hand and object are detected. However, the robot is not able to grasp the object because the conditions of the graspability are not

satisfied, e.g. when the height of the robot hand is greater than the distance between contact point of the object and human hand (safety factor for the human fingers during the grasping phase).

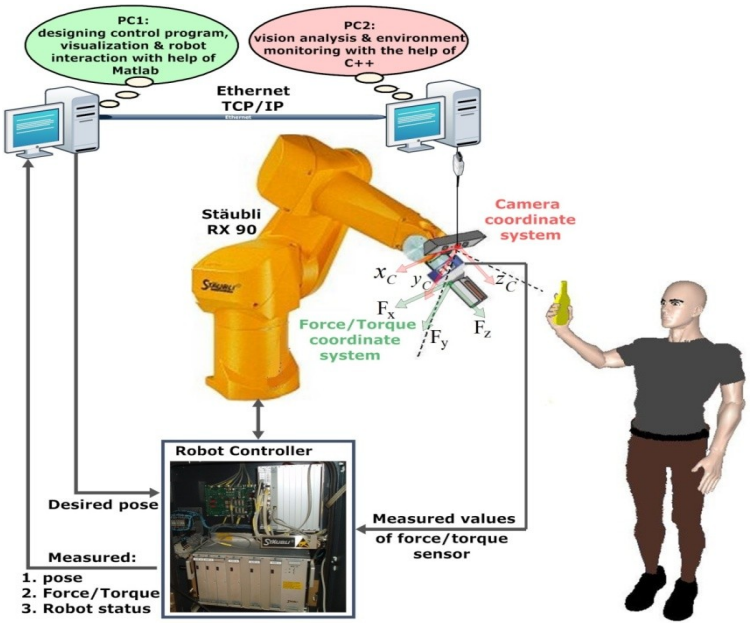

Fig. 5.35 Software & Hardware

5.3.1.2 Process algorithms

This section will explain the main procedures of the proposed system in order to perform the handing-over tasks from human hand to the robot hand.

Fig. 5.36 presents the algorithms of the process supported with a voice subsystem to tell the user about the current phase and the status of the operation if any error has been happened or if the phase has been successfully performed. In the first phase the robot system will detect and segment the target object (see Chapter 2). Is the first phase successfully performed the robot will start to track the target object in two directions, x and y. The tracking algorithm can be easily extended to three directions x, y and z. However, in this phase, a safety distance in z direction between the robot hand and the human hand should be always kept. The robot will not move toward the human hand in z direction to establish the contact until it ensures that the human hand doesn't move anymore. When the human hand is in a stable position and the robot has already tracked it successfully, the robot system will start moving toward the human hand in z direction. The situation now is as following: The robot hand is around the target object but until now there is no contact between them. Hence, the fourth phase will start by searching for the first contact point. The robot will move slowly depending on the vision information and at the same time it will monitor the values of the force sensor. If the measured forces have exceeded some threshold (robot hand has established the contact with the object), the robot will close its hand. The last phase is the interaction phase. Interaction

phase means that the robot has grasped the object but the human has not released the object yet. In other word, the transported object will be like connection bridge between human hand and the robot hand. In this phase the robot will be able to react to the motion of the human hand with the help of force control.

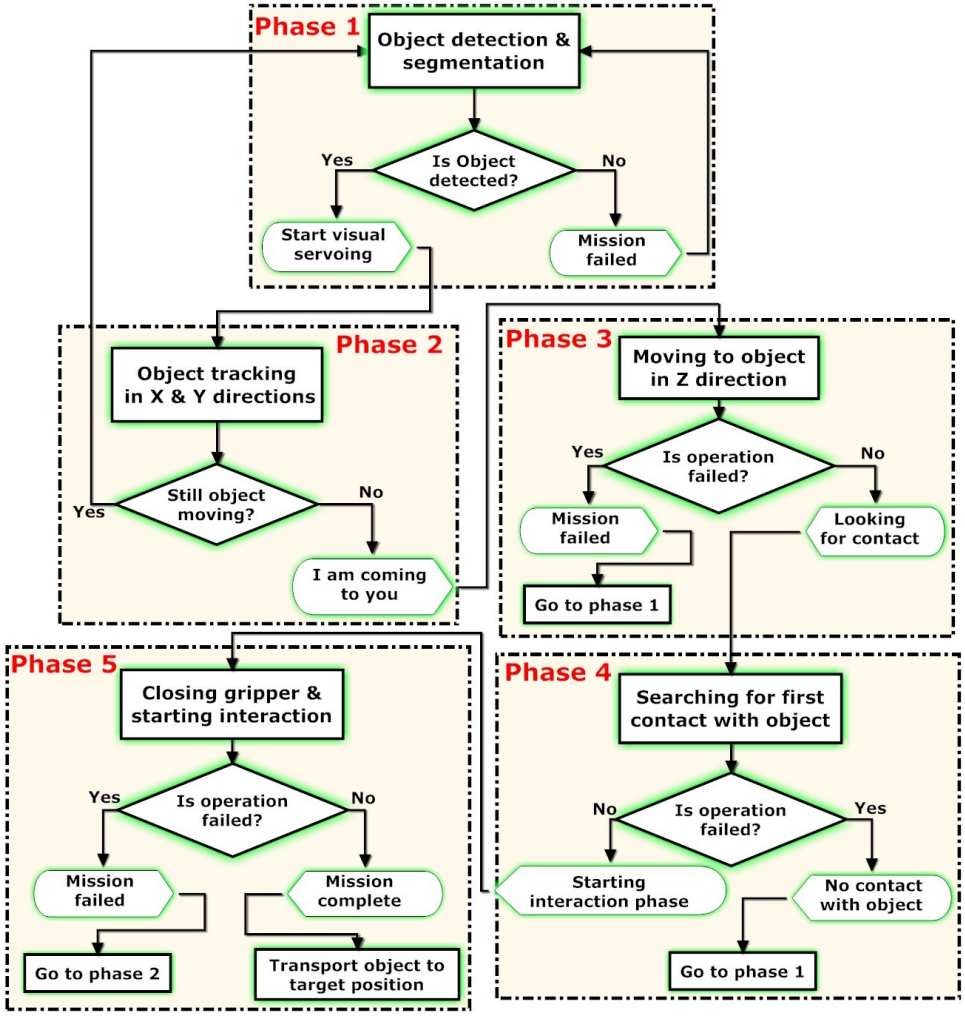

Fig. 5.36 Overview of process algorithms

At the end of image processing section, the system will send the value of $Vision_{stat}$ to the second PC depending on the image processing results as follows:

- When the segmentation result in the current frame is Okay and the characteristics of the segmented object (width and height) fit the characteristics of the robot hand. This means that the robot is able to grasp the object and the tracking phase will start (status is "Tracking" and $\text{Vision\_status} = 4$).
- When the segmentation result in the current frame is Okay and the characteristics of the segmented object (width and height) doesn't fit the characteristics of the gripper (e.g. $Obj_{height} < Grip_{height}$). This means that the robot is not able to grasp the object

(fingers will not be in a safety zone). The status will be "Not Able To Grasp" ($\text{Vision\_status} = 5$) and the robot will ask the human to grasp the object in different pose.

- When the target object has very similar color of the human skin (e.g. wood object). As previously explained (see Chapter 2), during the rotation the object color will be outside of the HSV ranges of the human-skin, even if the object has the same color of the human skin (light reflection effect). Hence, the segmentation result could be in the current frame not good (e.g. insufficient number of human hand points or the object is not detected), but in a previous frame (during human hand rotation) the segmentation result was Okay (status was "Tracking" and contact point was calculated). In this case, if the face in current frame is still detected, the system will lock the best results of the captured frames and the status will still "Tracking".
- When the segmentation result is not good (always the object is not detected) but the face is still detected. This could means that the human has empty hand. Hence, the status will be "Only Hand" and $\text{Vision\_status} = 3$.
- When all values of the mask $M(x, y)$ are zero (even inside the area-of-interest), this means that there is human hand or object in the current frame. The status will be "Face Only" and $\text{Vision\_status} = 2$.
- When the face is not detection, the frame will be skipped and the status will be "No Face" $\text{Vision\_status} = 1$.

In the next section the algorithms of the visual tracking of the carried object will be illustrated.

5.3.1.3 Human hand/carried object tracking in XY plane

As previously assumed, the human will be considered as a negative giver (bind, elderly or disabled) which is the most difficult case for the robot control system. This means, the human hand could move with the object in all directions at random. In this phase, the robot will track the human hand with the object only in XY plane in order to ensure the safety of the human, because by tracking the human hand with object in z direction could be dangerous especially when the human hand suddenly moves toward the robot. Where, z represents the distance between the robot hand and human hand with the object as shown in Fig. 5.37. Hence, a safety distance should be always kept in this phase between the robot hand and the human hand. The robot will track exactly the contact point of the object carried by the human hand which is calculated in Chapter 4. The robot will not move toward the human hand in z direction until it ensures that the human hand doesn't move anymore.

A real time tracking of moving object is not an easy task, especially when this object is moving randomly and the vision system cannot predict its motion. In our approach, the robot should track the loadfree hand or the object carried out by a human hand. Hence, the challenge in this phase is to design a control system which makes the robot able to track them smoothly and with sufficient speed in the real time in spite of the following difficulties:

- The position of the object always changes and it can move in all directions.
- The speed and acceleration of the object motion are not constant and the motion direction of the object could suddenly be changed.
- The cycle time of the vision algorithms for detecting loadfree or loaded hand could be a little changed from one frame to another depending on the scene.
- In this work, older commercial robot Stäubli RX90 is used, so the commercial controller cannot change the motion speed of the robot when the robot is moving. Moreover, it cannot change the desired position or send a new one suddenly, unless the robot reaches its current position or a "brake" instruction is given.

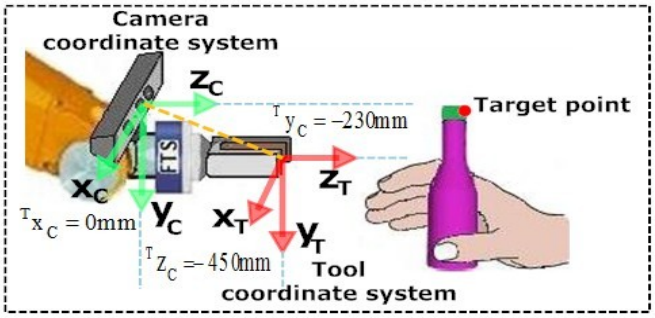

Fig. 5.37 Coordinate system of the experiment

Fig. 5.37 shows the camera coordinate system which locates at the middle of the Kinect camera and the tool coordinate system which locates at the end of the gripper. The camera is installed on the robot (camera in hand position) which means that the relation between camera coordinate system and tool coordinate system is constant. The relative position between camera coordinate system and tool coordinate system consists of translation part only in y and z axes and there is no rotation difference between the coordinate systems or translation in x axis ($^T x_C = 0$). With $^T y_C$ and $^T z_C$ will be further denoted the translation between camera coordinate system and tool coordinate system in y and z.

$$\begin{aligned} z_T &= z_C + {}^T z_C \\ y_T &= y_C + {}^T y_C \\ x_T &= x_C \end{aligned} \tag{5.44}$$

When $Vision_{stat} = 4$, the robot will start tracking the contact point of the object. The main purpose of the tracking phase is to preserve the target point of the object or the hand at the middle of the camera's view. (Cam_x, Cam_y, Cam_z) represents the position of the middle point of the camera. The required distance (mm) in order to locate the tracking point of the object at the middle of camera's view (position based visual servoing approach) is as follow:

$$\begin{aligned} Err_x &= Obj_{x_C} - Cam_x \\ Err_y &= Obj_{y_C} - Cam_y \\ Err_z &= Obj_{z_C} - Cam_z \end{aligned} \tag{5.45}$$

where ($Obj_{x_C}, Obj_{y_C}, Obj_{z_C}$) represent the tracking point of the target object or of the human hand depending on the task in (mm).

As is known, in some older commercial robots, the motion speed or desired position cannot be changed when the robot is moving, i.e. the robot is not able to receive a new target position unless the previous motion is finished. Hence, a normal position controller will not be sufficient, especially in the approach, when the object direction and position can suddenly be changed by human hand. Therefore, in order to track the object smoothly and with sufficient speed in real time, the proposed approach will not control the speed of the robot directly but it will control the motion steps of the robot. Whenever the target point is farther, the robot will move toward it with a greater step. The next section will illustrate the control system in x direction and the same algorithms will be used to control the y direction. The control equation in x direction is written as follows:

$$Cam_{d_x} = Cam_{m_x} + sgin\,(Err_x) * Step_x \qquad (5.46)$$

where Cam_{m_x} is the position of the camera's middle point in the world coordinate system which is measured by robot controller. Cam_{d_x} is the desired position of the camera's middle point in the world coordinate system. $sgin\,(Err_x)$ is the sign of the error value, which will define the direction of the motion. $Step_x$ is the motion step in x direction which will be illustrated in the next section.

As shown in Fig. 5.38 the visual tracking diagram is divided into five zones depending on the values of motion step $Step_x$. The value of $Step_x$ is related with the vision errors Err_x. Is the object is too far from the robot, the robot will move toward the object with accelerated large steps. The closer the robot is to the object, the smaller the motion step will be. In real experiment, even if the human wants to be the natural giver and doesn't move his/her hand completely, the human hand will not be stable, and it could moves one or two centimeters. Hence, to avoid this useless tracking, the visual tracking will not be required when the vision error is less than 2cm (zone 0) and this error will be corrected later using the force sensor.

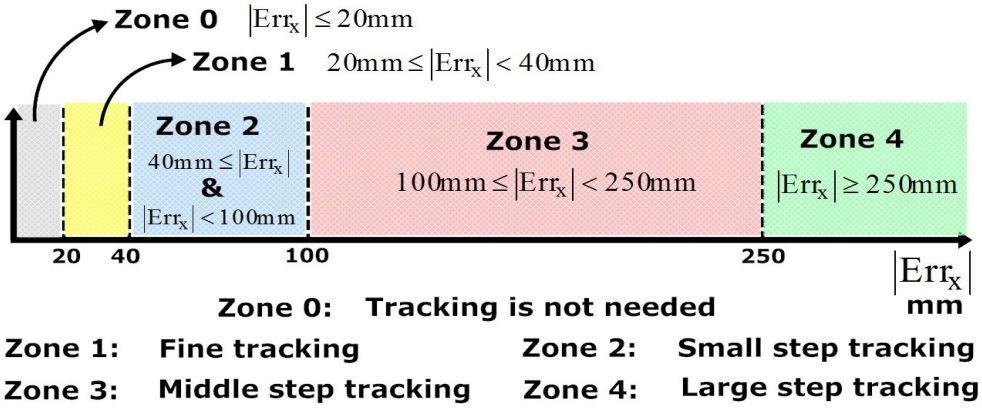

Fig. 5.38 Zones of the visual tracking

In every zone, $Step_x$ is limited by two limits ($MinRan_x$, $MaxRan_x$) and depending on them, the system will define if the steps value should be increased, decreased or constant. In brief, depending on the error value Err_x, the motion step $Step_x$ will be updated in every cycle time and it can be in three different cases:

- Accelerated steps: As previously mentioned, the human could be negative giver (human hand is moving away from the robot), which means that even if the robot is moving toward the object, the error Err_x could still increase because of the human motion. In this case the robot should move toward human with the accelerated steps:

$$\text{If } (Step_x < MinRan_x) \; then \; (Step_x = \; Step_x + Acc_x) \qquad (5.47)$$

- Decelerated steps: When the error Err_x is becoming smaller, which means that the robot can track the human hand and it can be closer to object. In this case the system will decelerate the motion step of the robot in order to get smooth tracking without overshooting the target:

$$\text{If } (Step_x > MaxRan_x) \; then \; (Step_x = \; Step_x - Dec_x) \qquad (5.48)$$

- Constant steps: In any zone, when the value of $Step_x$ is inside the ranges limits ($MinRan_x$, $MaxRan_x$), the motion step will be constant.

$$\text{If } (MinRan_x < Step_x < MaxRan_x) \; then \; (Step_x = \; Step_x) \qquad (5.49)$$

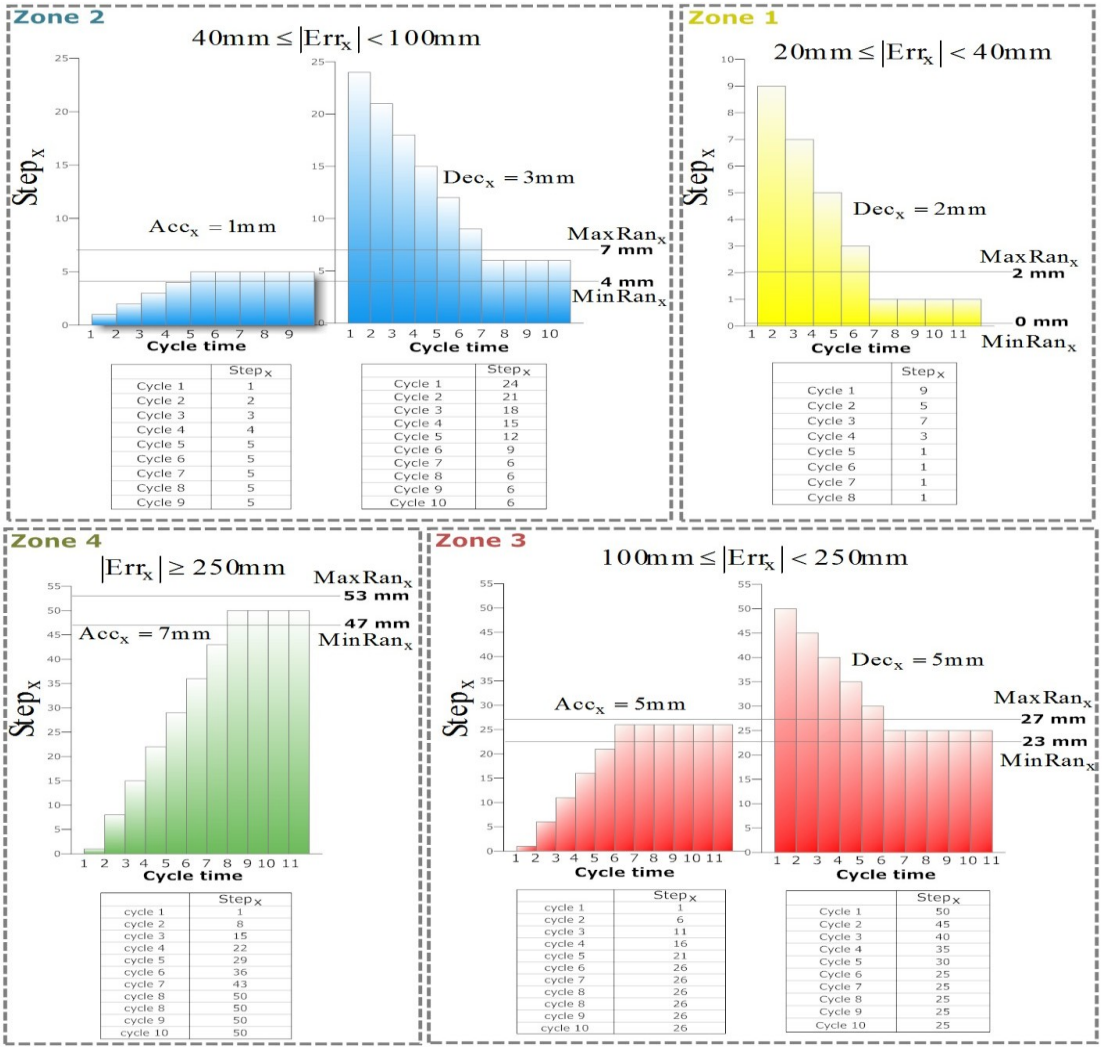

Fig. 5.39 Motion step in different zones

Fig. 5.39 presents the changes of the motion steps in all zones. When the robot is very near to the object and the error value is inside the zone1, e.g. Err_x is less than 4cm, the motion step will be either constant or decelerated step. If $Step_x$ is inside the ranges $(MinRan_x, MaxRan_x)$, $Step_x$ will be held constant until the robot reaches the target point. Whereas, if the robot is coming from distant point closer to the target point which means that Err_x is decreasing and it could move from one zone to other one, e.g. from zone2 to zone1, in this case the limits $(MinRan_x, MaxRan_x)$ of $Step_x$ will be changed in order to fit the motion in the new zone. Hence, $Step_x$ will be decreased (decelerated steps) until its value becomes inside the range $(MinRan_x, MaxRan_x)$ of the new zone. In any zone, when the value $Step_x$ is inside the range $(MinRan_x, MaxRan_x)$, the motion step will be constant.

In general, Err_x when moving from one tracking zone to a higher tracking zone, the values of the ranges $(MinRan_x, MaxRan_x)$ will be larger in order to fit the new zone and $Step_x$ will be increased (as shown in Equation (5.47)) until its value comes inside the range of the new zone, after that the value of $Step_x$ will be constant (as shown in Equation (5.49)) until the zone is changed or the robot reaches the target point. This case could be happened, when the robot is very close to the target point, e.g. Err_x inside the zone1. If the human hand has moved his/her hand suddenly away from the robot, the value of Err_x will suddenly increase and may come inside the zone4. Motion step will react to that motion and it will increase in order to let the robot to track the target point with sufficient speed. In the opposite case, when Err_x moves from one tracking zone to a lower tracking zone, the values of the limits $(MinRan_x, MaxRan_x)$ will become smaller in order to fit the new zone and $Step_x$ will be decreased (as shown in (5.47)) until its value comes inside the new range, after that the value of $Step_x$ will be constant (as shown in Equation (5.49)) until the zone is changed again or the robot reaches the target point. This case could be happened, when the robot is becoming closer and closer to the target point while the human hand behavior is as natural giver (doesn't move). Now the values of Err_x and $Step_x$ will gradually decrease until the robot reaches the target point with smooth motion. The proposed tracking algorithm estimates the speed of the human hand indirectly by calculating the variations and changes which have occurred to the human hand location between every two frames. During the experiments, the proposed approach has proved high ability to react rapidly to all changes in speed, direction and position of target point and it can be used in any application which needs to track a moving object visually, as shown in Fig. 5.40, Fig. 5.41. The values of $(Step_x,\ MinRan_x,\ MaxRan_x, Acc_x, Dec_x)$ and the number of zones will depend on the type of application. In this work, they have been optimized during the experiments.

Fig. 5.40 and Fig. 5.41 show the experimental results of the proposed visual tracking approach in Y and X directions. The green diagram presents the visual error Err_x (the difference between the target point of the object or hand and the camera's middle point) in the camera coordinate system. Whereas the red diagram presents the motion of robot in the world coordinate system. The diagrams are divided into different tracking zones which mentioned previously in Fig. 5.38 and Err_x is moving between them. From the behavior of robot motion

(red diagram), it is clear that the robot always moves toward the target point as positive party therefore the visual error is diminished to zero. On the contrary, from the behavior of the human it is clear that the human increases the visual error extremely (as negative giver) by moving his/her hand, look e.g. at time t = 10s in Fig. 5.40).

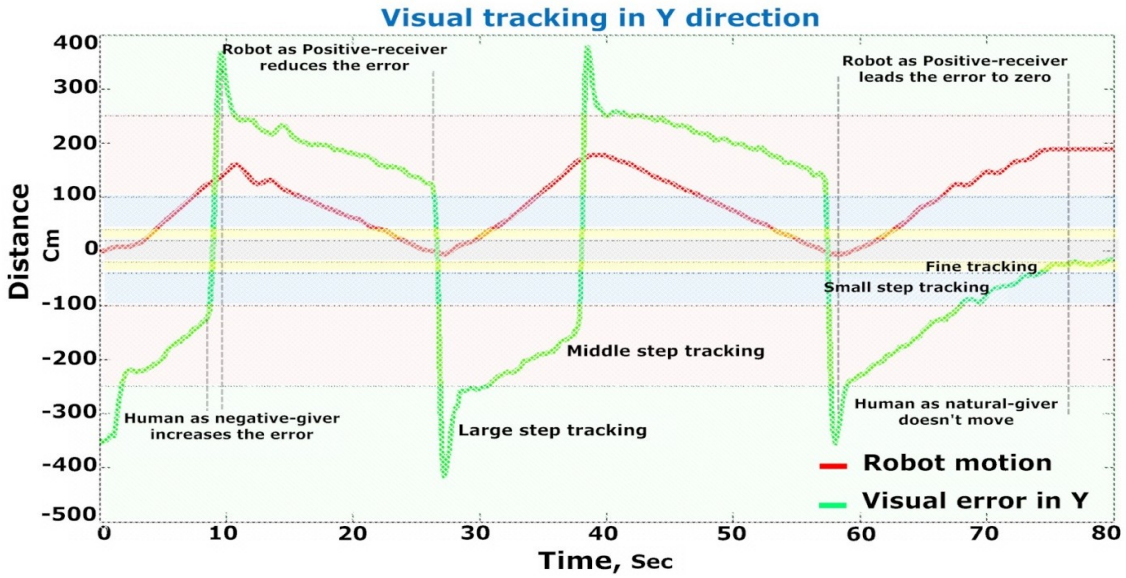

Fig. 5.40 Visual tracking in Y direction

At time t = 0s in Fig. 5.40, the visual error was near to -350cm which made the robot moves upwards to reduce the visual error. Starting in time t = 0s, until t = 2s, the visual error was in inside the large step tracking zone and it has diminished. At time t = 2s, the visual error has come inside the middle step tracking zone (which means $Step_x$ should be reduced as shown in Fig. 5.39). However, at time t = 8s the human hand has suddenly been moved and it has changed the visual error from -120cm to 380cm which has led the robot to react rapidly and to change the direction of the motion in order to track the target point. The rapid visual tracking and changing directions will be continued until the human hand acts at least as a natural giver (e.g. at time t = 58s), in this case the robot will be still moving toward the target point smoothly and the visual error will be diminished until it enters the zone0 (visual error is less than 2cm). It means that the robot has arrived to the target point, after that the visual tracking will be stopped and next phase will be started.

The same approach will be applied to track the target point in X direction as is shown in Fig. 5.41, At time t = 0s the visual error was near -250cm which made the robot move upwards to reduce the visual error. At time t = 13s the human hand has suddenly moved and changed the visual error from -103cm to 388cm. This has led the robot to react rapidly and to change the direction of the motion in order to track the target point. The rapid visual tracking and changing of directions will continue until the human hand acts at least as a natural giver (at time t = 59s). In this case the robot will still be moving toward the target point smoothly until it enters the zone0.

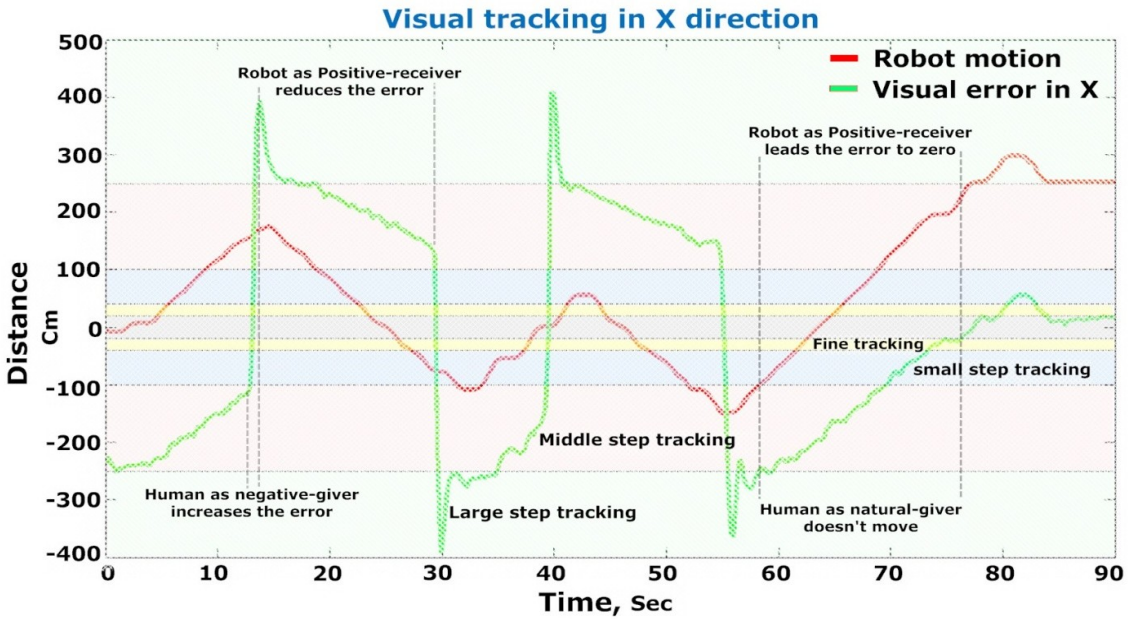

Fig. 5.41 Visual tracking in X direction

When the visual tracking phase is successfully performed, the robot system will start moving toward the object or the loadfree hand in z direction in order to establish the first physical contact with them. This phase will be illustrated in the next section.

5.3.1.4 Moving to the object and searching for first contact

In this phase the robot will move toward the object or the human hand in z direction in order to arrive to the contact point and then to establish the first contact with it. This motion will be monitored by the force sensor to ensure the safety. Furthermore, as previously explained, even if the human serve as natural giver and doesn't move his hand, the human hand will not be stable, and it could moves one or two centimeters. This means, that visual tracking could have a small vision error (less than 2cm as is shown in zone 0 in Fig. 5.38). This error will be corrected later using the force sensor, i.e. the fine tracking for the contact point of the object will be performed by the force sensor. When the visual tracking phase is finished and the robot arrived to the point (x_O, y_O, z_O), the robot will move as additional distance (20mm) in z direction toward the object before establishing any contact with the object. After that the robot will start searching for the first physically contact with the help of the force sensor. In this way, the robot will ensure that it will grasp the object successfully and then it will be able to close the gripper fingers on the object.

Fig. 5.42 shows experimental results where the requirements of the force safety factor have been fulfilled. In this experiment, the robot is moving toward the human hand to grasp the object in z direction (at t = 0s, z = -4.6cm and then at t = 27,5s, z = 29.7cm). Robot has arrived to the target point (tracking point) of the object at t = 27,5s without any unexpected obstacles, so that in the next phase at t = 28s robot will start searching for the first contact with the object. As shown in Fig. 5.42, robot will move slowly in x direction (at t = 28s, x = 9.12cm and

then at t = 33s, x = 7.45cm). Hence, when the applied force in x direction exceeds the desired threshold (e.g. 3N is high enough so that the robot system can recognize contact with the object). In this case robot system will close the robot hand in order to grasp the object.

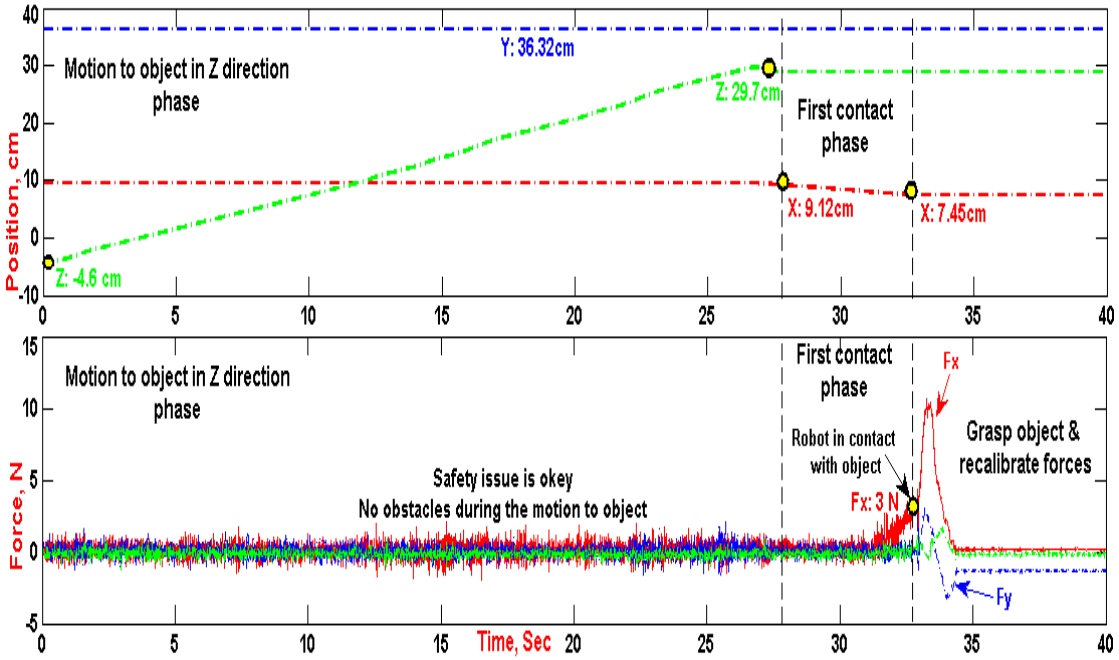

Fig. 5.42 Moving toward the object and searching for first contact

An important question in this section is how the robot can recognize that e.g. the applied force in x direction will serve as contact force with the object and the applied force in z direction will be impact force with an unexpected obstacle. Answer to this question will be performed with the help of the automatic decision system (see Chapter 4). Briefly, with the help of vision system, the robot system will define when and in which direction it should apply the contact force on the target object, so that any other measured force in an unexpected time or in different direction will be the warning sign and the force safety factor will be activated.

5.3.1.5 Interaction phase

Interaction phase will start when the robot has already grasped the object but the human has not released the object yet. In this case, the target object, which will be transported from human hand to the robot hand, will serve as a connection bridge between both. In this phase, we will assume that the human may not release the object immediately. Perhaps, the human would like to drive the robot toward another place, so the robot should be able to react to the motion of the human hand.

Fig. 5.43 presents the interaction phase when the human and robot grasp the object at the same time. In this phase, the human will react as master and the robot will be the slave. Robot will react to the motion of the human hand with the help of force control.

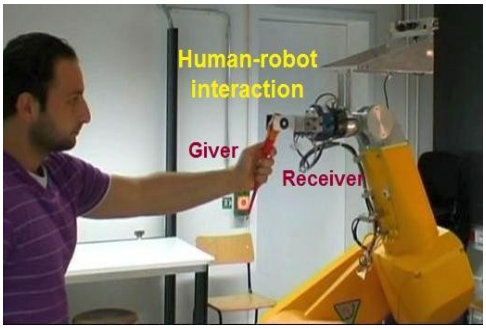

Fig. 5.43 Interaction phase

Fig. 5.44 illustrates experimental measurements of the forces and positions of the robot during the interaction phase. It is divided into three diagrams (three diagrams for three directions x, y and z). Every diagram contains two curves: the dash-point curve represents the position values in one direction (in first diagram in direction x) and the solid curve represents the force values in the same direction.

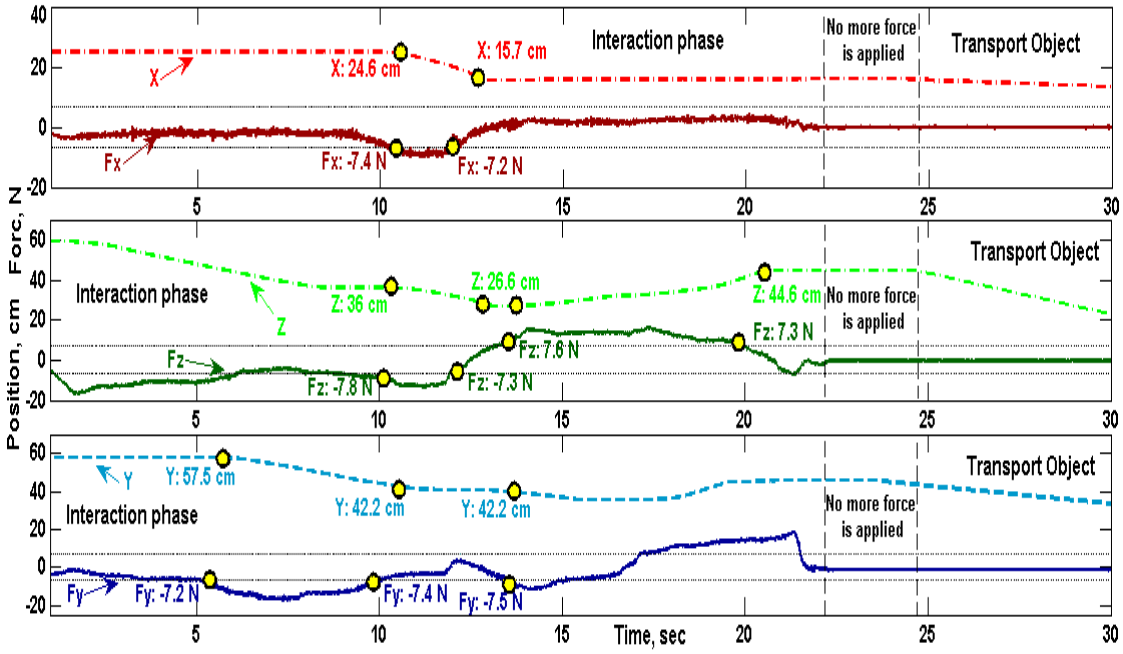

Fig. 5.44 Results during interaction phase

Fig. 5.44 illustrates how the robot reacts and moves according to the forces which are applied to the robot by the human hand. As long as the applied forces are inside the threshold range (-7N < F < 7N), robot will not move. Whereas, as shown in the first diagram in time t =10s, when the applied force in x direction is smaller than threshold -7N (F_x = -7.4N), the robot will start moving in the negative x direction (the robot has moved from position x = 24.6cm to x = 15.7cm). Robot will still move until the forces applied by human return inside the threshold range. The same algorithms will be applied in directions y and z. As shown in the second diagram in time t = 13s, the applied force in z direction is greater than the threshold 7N (Fz =

7.6N). In this case, the robot will move in positive z direction until the applied force returns to the threshold range (robot has moved from position z = 26.6cm to z = 44.6cm). The interaction phase will still be active until the human stops applying any force in any directions, i.e. the human has already released the object. After that the robot will wait for a few seconds and then it will transport the object to the target position or elsewhere. Otherwise, by transfer of object to human hand, robot will release the object when the human pulls the object with desired force and after that it will return home.

5.3.1.6 Safety procedures

In reference (Haddadin, Albu-Schäffer, Strohmayr, Frommberger, & Hirzinger, 2008) and (Haddadin, Albu-Schäffer, & Hirzinger, 2007), Haddadin has evaluated the injuries which could happen during the human robot interaction relating to the robot speed, robot mass and constraints on the environment. A lot of papers have been published which have proposed different solutions for improving the safety factor during the interaction between human and robot. (Bicchi & Tonietti, 2004) and (Zinn, Khatib, & Roth, 2004) e.g. have proposed to improve the mechanical design of the robot by reducing the mass of the robot, where another work (Mainprice, Sisbot, Simeon, & Alami, 2010) has proposed trajectories which consider constraints related to the human body. However, our work will not focus on robot design or trajectories plan to improve the safety factor, instead it focuses on the benefits of using and integrating the vision and force sensors in order to improve the safety factor during the physical interaction between human and robot. In our opinion, even if the system would use lightweight robot and predefined trajectories, it is indispensable of integrate vision and force information to guarantee the safety especially when unexpected problems or errors happen during the physical interaction.

This section will illustrate how the proposed procedures are performed to ensure the safety with the help of vision and force information and it will present proposed voice subsystem which will help to increase the safety of the user especially if the user is blind.

Vision procedures for safety

This section will propose two vision safety factors. The first safety one (SF_body) will be related to guarantee the safety of the whole human body, whereas the second safety factor (SF_hand) will be related only to the safety of the fingers during handing-over the object. Values of both factors will be zero as long as the safety requirements are fulfilled. Otherwise, if any error or dangerous position of human is recognized, the safety variables will be immediately activated and the task will be canceled.

As is shown previously in Chapter 3, face detection algorithm is implemented. Robot system can detect the human face only when the human looks directly to the camera or with deviation of up to ± 50°. When the robot system can detect the human face, this means that the robot is also within the sight view of the human and the human can see the motion of the robot. Face detection could be considered as positive sign which helps the robot to recognize if

the human is able to follow its movements and is prepared to react. In the case of blind user, the voice subsystem of the robot will help the user to recognize the robot's direction, so the user will have to move his head toward the robot.

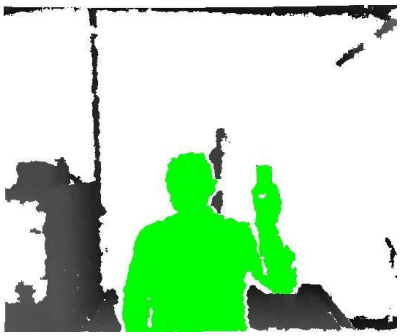

Fig. 5.45 Human body segmentation

Fig. 5.45 presents the results of segmentation of the human body and of the related target object carried by human hand. This segmentation step will follow before distinguishing between the object from the human hand. Human body segmentation will help the robot system to define the depth map of the body. With the help of human body depth map, robot system can detect if any part of the human body expects the active hand (the hand which carries the target object) is located near area of interest. In our procedures the area of interest which contains the target object and the active hand should be the nearest part of the human body to the robot, and other parts of the human body should be located far away from the area of interest (80mm) as shown in Fig. 5.46.

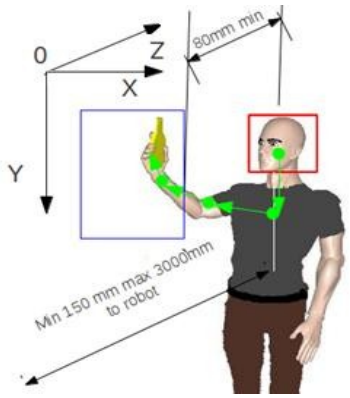

Fig. 5.46 Area-of-interest

Hence, SF_body will be activated and the task will be canceled in the following cases: If the robot system is not able to detect the human face during the task anymore, if any part of the human body is closer than the active hand to the robot or if any part of the human body such as head or chest is located in distance less than 80mm from the area of interest (in the depth map). If more than one person are inside the sight view of the robot system, it will handle and interact only with the closest person and all other users will be ignored. If the closest person has no objects in his/her hand, the operation will not be performed.

The second safety factor (SF_hand) will be needed during the first contact phase between human hand and robot hand, when the robot moves toward the human hand to grasp the object. As shown in chapter 4, robot system will calculate the graspability by defining the boundary line between human hand and object. When the robot system defines the graspability, it will compare between the characteristics of robot hand and the size of the object (width and length). During this phase the robot system will add a safety zone to protect the fingers of the user. If the user carries the object in a way that the robot will not be able to grasp it, the safety factor of hand will be activated and the mission will be canceled.

Force procedures for safety

In (Reinhart, Zaidan, & Hubele, 2010), the authors have presented the importance of monitoring and controlling the force information, especially in cooperative systems and motions guided by human. The proposed procedure in this work will include one force safety factor (SF_force). The value of this factor will be zero as long as the force safety requirements are fulfilled. However, if any errors or unexpected values of force are recognized, the safety factor will be immediately activated and the mission will be canceled. Monitoring the force values is very important, especially when the robot is moving toward the human (z direction). In this phase the speed of the robot could be fast which means that a hard impact force could arise if any unexpected obstacle has appeared or if the human has moved toward the robot in an unexpected way.

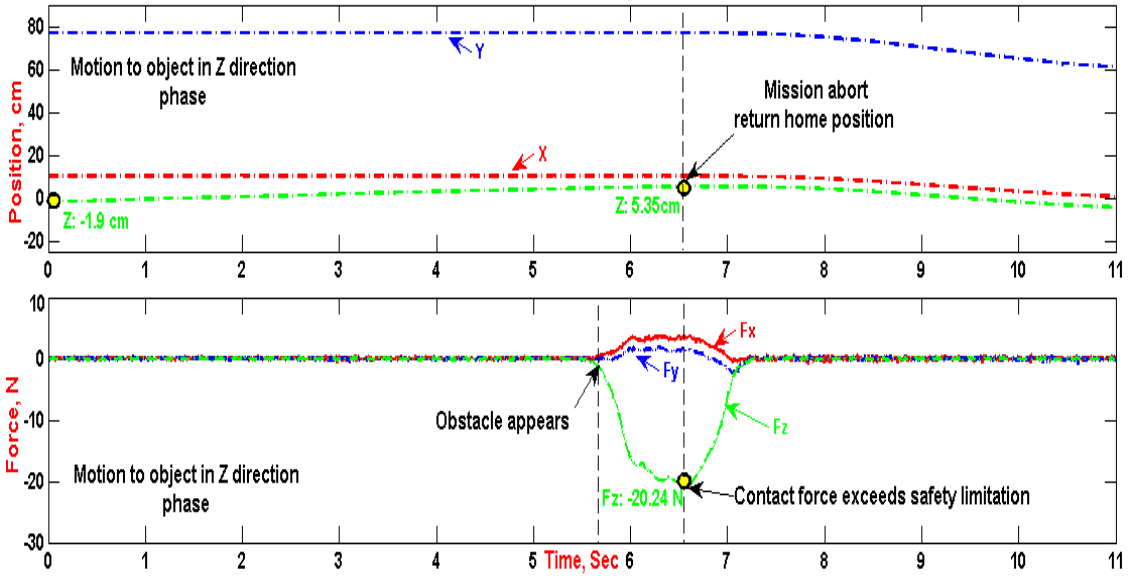

Fig. 5.47 Aborted mission because of Safety-issue

Fig. 5.47 illustrates how the robot system will react if any unexpected force is measured, especially when the robot is moving toward the object in z direction. As shown in Fig. 5.47, the measurement starts when the robot is moving toward the human to grasp the object. The initial position in z direction was (z = -1.9cm). During this phase, unexpected obstacle has faced the robot at t = 5,7s, so that the applied force in negative z direction will be increased. When

the measured force exceeds the safety limit (SL = ± 20N), the force safety factor will be immediately activated and the task will be canceled. As shown in Fig. 5.47, at t =6,6s the measured force in z direction has exceeded the safety limit Fz = -20.24N, so that the task will be immediately aborted and the robot will return back.

Voice subsystem

Robot system is supported with voice subsystem. It will announce the current phase (what it is going to do), the status of the operation or if some errors have occurred. The voice subsystem will give the human the opportunity to know and to understand what the robot is doing now and to be ready if any error has occurred during the task. It will increase the safety factor between human and the robot, especially if the user is disabled or blind.

Event	Status	Voice message
New face detected	FACE DETECTED	Nice to meet you
Person leaves frame	NO FACES	Good Bye
Person detected but distance is too big	FACE DETECTED	Please come closer
Person detected, object not found	HAND ONLY	Hand only detected
Distance O.K. object segmentation successful	TRACKING	Object size in mm. I am tracking
Vision phase complete, robot began moving to object	COMING PHASE	I am coming to you
Moving failed	COMING PHASE BREAKS	I can't come to you, Operation failed
Searching first contact point with force sensor	SEARCHING CONTACT	Searching for contact
Starting force interaction	INTERACTION	Starting interaction phase
Failing interaction phase	INTERACTION BREAKS	No contact with object
Interaction completely successful	NO MORE FACES	Return to home position

Table 5.2 Status of voice subsystem

The voice subsystem can be easily improved in order to announce different states. Table 5.2 illustrates only the main status of the proposed tasks (transferring objects between human and robot). This version of voice subsystem is suitable for our experiments. If a mobile robot were used, the voice messages could be easily modified.

5.3.1.7 Control algorithms

This section will discuss the structure of the proposed control system and the algorithms of fusing vision and force control which are inspired from the proposed control structure in

chapter 4. Fig. 5.48 illustrates the control algorithms which combine vision and force feedback. With X or $\vec{X}$ will be further denoted the pose of end-effector in different coordinate systems. With C, W and E the coordinate systems of camera, world and end-effector are denoted. Here ${}^{E}\vec{X}_{m}$ is the measured pose of end-effector which comes from robot control system, ${}^{E}\vec{X}_{V}$ is desired pose of end-effector which comes from vision. Poses are transformed from camera coordinate system and world coordinate system to end-effector coordinate system using transformation matrixes ${}^{E}T_{C}$ and ${}^{E}T_{W}$.

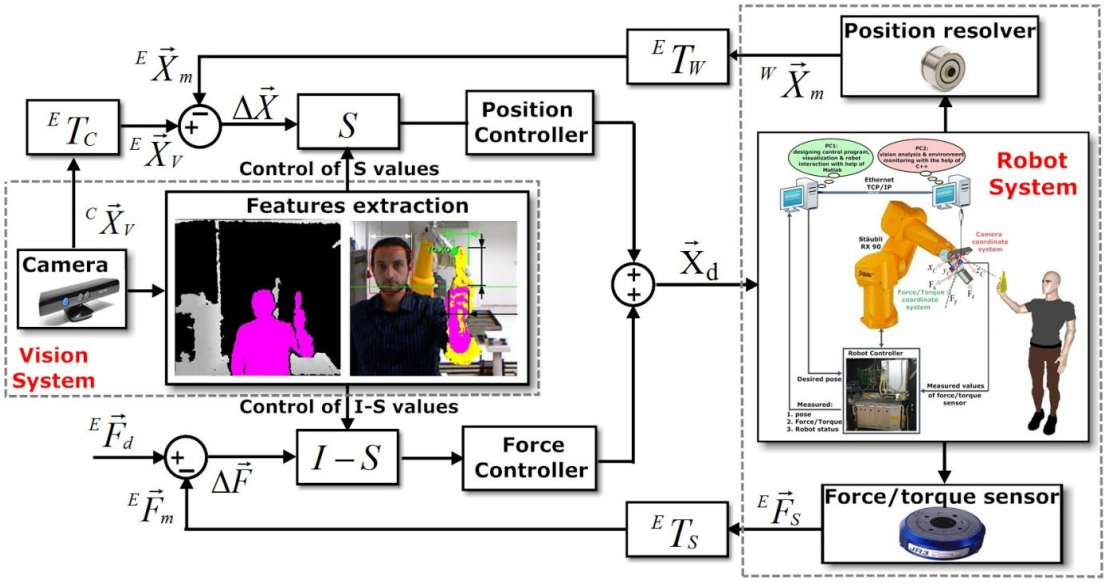

Fig. 5.48 Control algorithms

As previously explained, the robot system will find out (selection matrix values S, S_i = 1 or 0) which subspace will be force controlled ($\Delta\vec{F}$) and which one will be position controlled ($\Delta\vec{X}$):

$$S\cdot\Delta\vec{X} = \begin{bmatrix} S_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & S_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & S_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & S_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & S_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & S_6 \end{bmatrix} \cdot \begin{bmatrix} P_x \\ P_y \\ P_z \\ \psi \\ \vartheta \\ \varphi \end{bmatrix} = \begin{bmatrix} P_{xs} \\ P_{ys} \\ P_{zs} \\ \psi_S \\ \vartheta_S \\ \varphi_S \end{bmatrix} \qquad (5.50)$$

$$\Delta\vec{X} = {}^{E}\vec{X}_{V} - {}^{E}\vec{X}_{m} \qquad (5.51)$$

$$(I-S)\cdot\Delta\vec{F} = \begin{bmatrix} 1-S_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1-S_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1-S_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1-S_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1-S_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1-S_6 \end{bmatrix} \cdot \begin{bmatrix} \Delta F_x \\ \Delta F_y \\ \Delta F_z \\ \Delta M_x \\ \Delta M_y \\ \Delta M_z \end{bmatrix} = \begin{bmatrix} \Delta F_{xs} \\ \Delta F_{ys} \\ \Delta F_{zs} \\ \Delta M_{xs} \\ \Delta M_{ys} \\ \Delta M_{zs} \end{bmatrix} \qquad (5.52)$$

where $_{\Delta}\vec{F}$ (6x1) is the force error computed from the desired force ${}^{E}\vec{F}_{d}$ vector and measured force vector ${}^{E}\vec{F}_{m}$:

$$\Delta\vec{F} = {}^{E}\vec{F}_{d} - {}^{E}\vec{F}_{m} \tag{5.53}$$

$$\Delta\vec{F}_{S} = (I - S)\cdot\Delta\vec{F} \tag{5.54}$$

5.3.1.8 Conclusion

Some experimental results and diagrams have been already presented. Whereas, Fig. 5.49 presents further experimental results for the whole handing-over scene from the human hand.

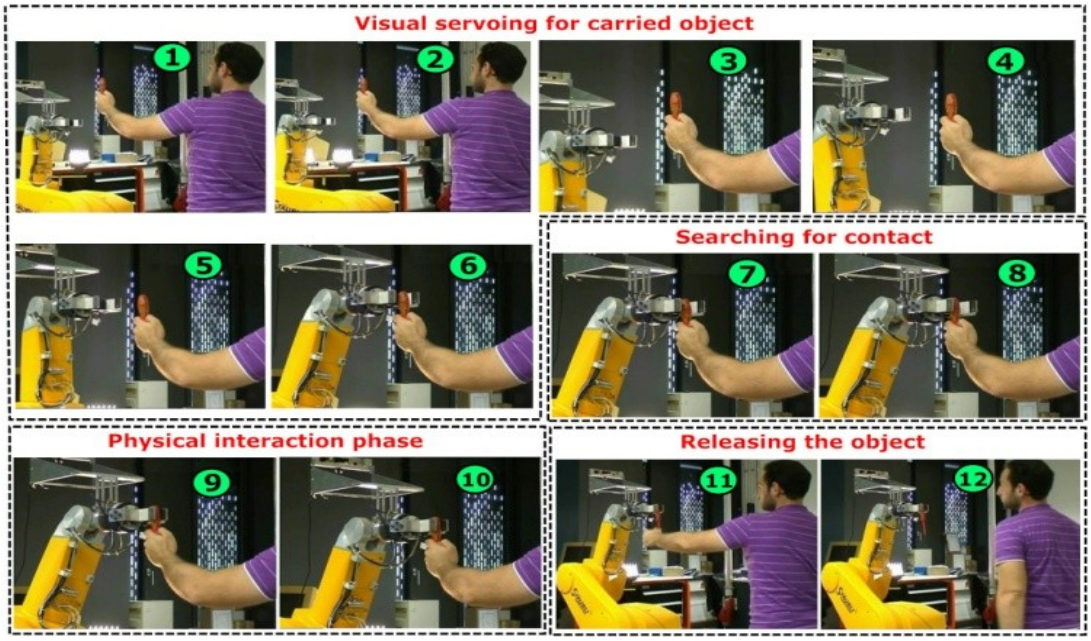

Fig. 5.49 Handing-over object from human hand

In Fig. 5.49, as shown in the first six pictures, the robot is tracking and moving toward the object carried by human hand in order to grasp it from the human hand with the help of vision system. When robot reaches the tracking point of the object, the robot will start searching for the first contact point with the object using force reading, as shown in pictures 7 and 8. When the force sensor measures that the desired contact with the object has been established, then the robot will close its hand to grasp the object as shown in pictures 9 and 10. Hence, the physical interaction between the human and the robot will start and the robot will react to the motion of the human hand on the basis of force sensor measurements. When the interaction phase is finished, the human will release the object as shown in pictures 11 and 12.

In conclusion, this section has proposed an automated robot system which is able to grasp any object from the human hand with totally no information about the model of the object. Vision and force integration will ensure the safety and the successful execution of the task.

5.3.2 Transferring model-free object to human hand

This section will illustrate the algorithms of combining visual servoing and force control in order to handing over model-free objects from undefined place to the human hand. This part of work will consider the following procedures:

- Robot should ensure the safety of the human during the interaction.
- Robot should define the position of loadfree human hand exactly.
- Robot should also learn if the human is ready to receive the object or not.
- Robot has to ensure that the object has been transferred to the human.

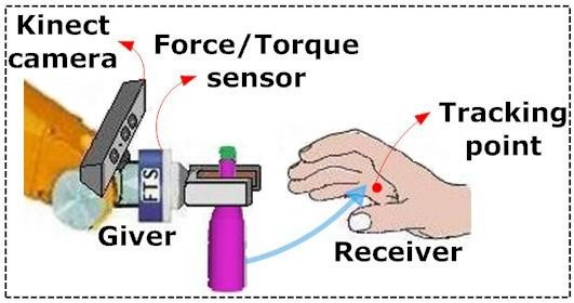

Fig. 5.50 Handing-over from Robot to human

Fig. 5.50 illustrates the moment of the handing-over between robot and human hand using vision and force feedback. The goal is to grasp the target object which is located on a table or conveyor and then to transport this object to the human hand. The experimental equipments are already explained in the previous section. In the next section, the algorithms of whole process will be presented

5.3.2.1 Process algorithms

Fig. 5.51 illustrates the main phases of the process. The vision system will help the robot to detect, segment and distinguish the target object from the other objects which are located nearby target object (see Chapter 2). After that the system will calculate the position and orientation of the object in world coordinate system.

In the following, robot will move toward the object to grasp it. Having grasped the object successfully, robot system will be ready to deliver it to the human hand. Hence, the robot system will start searching for human hand with the help of the OpenNI library and NITE toolbox. After detection of human hand the robot will track it in x and y directions with the help of visual servoing. The visual servoing phase for the human hand will be active until the human hand stop moving. When the human hand is stable and doesn't move anymore, robot will move toward it in z direction. In the last phase, robot will deliver the object to human hand with the help of force control. When the human touches the object, robot will measure the generated forces during the contact between human and robot. Will the force exceed desired threshold value, this means that the human has grasped the object, robot will open its gripper and release the object.

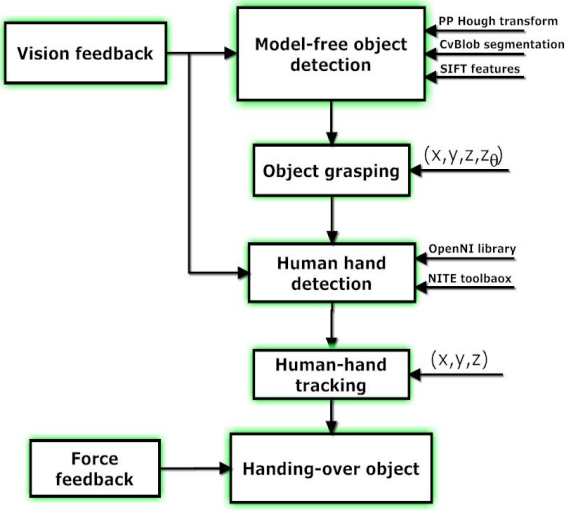

Fig. 5.51 Processing algorithms

Fig. 5.52 presents real results of the proposed vision algorithms to segment and to detect the target object, these algorithms are previously explained in Chapter 2, so the next section will explain how the robot will grasp the object depending on the vision information

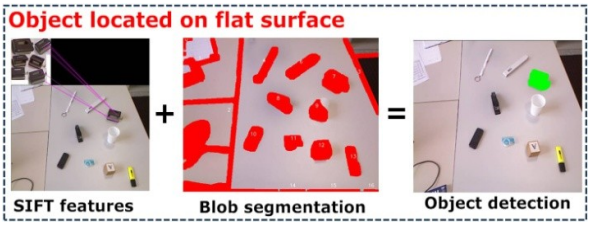

Fig. 5.52 Object segmentation and detection

5.3.2.2 Object grasping

Fig. 5.53 presents different objects which locate on a table. According to the results of image processing, the robot will detect the target object.

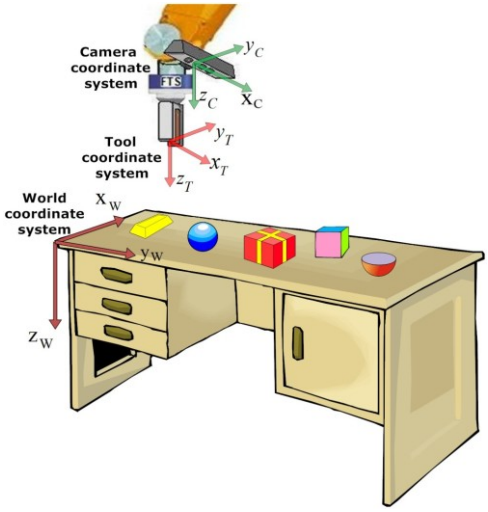

Fig. 5.53 Experimental environment

Fig. 5.54 presents the overview of the objects and the table from the camera perspective. The vision system will segment all the objects, and then the vision system will give every object will ID number. The next step will be defining the target object which has the most number of detected SIFT features.

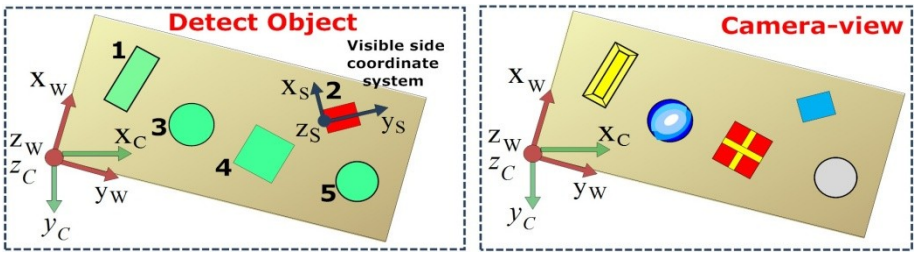

Fig. 5.54 Object grasping

As previously shown, using CvBlob library the system can extract some features of the blob (segment) i such as centroid, length, width, minimum and maximum pixels of the object etc.

$$Blobs.GetNumber(i, CBlobGetXCenter);$$
$$Blobs.GetNumber(i, CBlobGetLength); \quad (5.55)$$
$$Blobs.GetNumber(i, CBlobGetWidth);$$

Equation (5.55) presents the functions in CvBlob library which calculate the middle-point, length and width of the object. Using this information, the robot system can calculate the graspability (if the robot system is able to grasp the object or not). The graspability will be calculated (as shown previously in chapter 4) by comparing the width and length of the object with the width and object of the robot hand.

$$\begin{aligned} Max\_Vector[i].u &= Blobs.GetBlob(i)->MaxX(); \\ Max\_Vector[i].v &= Blobs.GetBlob(i)->MaxY(); \\ Min\_Vector[i].u &= Blobs.GetBlob(i)->MinX(); \\ Min\_Vector[i].v &= Blobs.GetBlob(i)->MinY(); \end{aligned} \quad (5.56)$$

Equation (5.56) presents the functions in CvBlob library which calculate the maximum and minimum pixels of the object in order to define how the robot will grasp it.

As shown in Fig. 5.54, the relative position between camera coordinate system and tool coordinate system consists only of translation part and there is no rotation difference between both coordinate systems. With ${}^{T}T_{C}$ will be denoted the translation matrix between the camera coordinate system and tool coordinate system. Whereas ${}^{C}T_{S}$ will be further denoted the transformation matrix between the visible side coordinate system and camera coordinate system (translation in x, y, z and rotation about θ_Z). If we assume that P_C, P_T and P_S are the coordinate of point P relative to camera, tool and visible side coordinate system, the transformation from one frame to other can be performed as follow:

$$P_T = {}^{T}T_{C} \cdot P_C \quad (5.57)$$

where ${}^{T}T_{C}$ can be written:

$$P_T = \begin{bmatrix} 1 & 0 & 0 & {}^{T}x_C \\ 0 & 1 & 0 & {}^{T}y_C \\ 0 & 0 & 1 & {}^{T}z_C \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot P_C \tag{5.58}$$

The transformation between the visible side coordinate system and the camera coordinate system will be performed as follow:

$$P_C = {}^{C}T_S \cdot P_S \tag{5.59}$$

where ${}^{C}T_S$ can be also written:

$$P_C = \begin{bmatrix} cos\theta_z & -sin\theta_z & 0 & {}^{C}x_S \\ sin\theta_z & cos\theta_z & 0 & {}^{C}y_S \\ 0 & 0 & 1 & {}^{C}z_S \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot P_S \tag{5.60}$$

By substituting (5.60) in (5.58), the robot system will calculate the direct transformation matrix between the visible side and the tool coordinate system.

$$P_T = \begin{bmatrix} 1 & 0 & 0 & {}^{T}x_C \\ 0 & 1 & 0 & {}^{T}y_C \\ 0 & 0 & 1 & {}^{T}z_C \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} cos\theta_z & -sin\theta_z & 0 & {}^{C}x_S \\ sin\theta_z & cos\theta_z & 0 & {}^{C}y_S \\ 0 & 0 & 1 & {}^{C}z_S \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot P_S \tag{5.61}$$

Hence, the robot system can move toward the target object and grasp it depending on the visual information.

5.3.2.3 Detection of loadfree human hand

When the grasp task has been performed successfully, the robot will start to search for human hand also with the help of vision feedback. In this task (handing-over object to the human hand), the human hand will be open and free on the contrary of the previous task (handing-over object from human hand). In other words, the human hand doesn't carry out any object; therefore the detection of open free human hand will be easier than the detection of human hand which is carrying undefined objects. In this task, the vision system can also use the model of the human body which is supported by NITE toolbox of the Kinect camera.

Fig. 5.55 Human hand detection

The rest of the proposed algorithms (human hand visual servoing phase and handing-over phase) to perform the transferring task to the human hand are similar to the proposed algorithms which are previously explained in transferring task from the human hand. The main difference between both tasks that in the first task the robot should grasp the object from the human hand, whereas in the second one the robot will release the object when the human grapes the object. Hence, here the robot system will move toward the human hand and wait until the human closes his hand on the object. When the force value exceeds desired values, this means that the human has already grasped the object, so the robot will release the object.

5.3.2.4 Conclusion

Fig. 5.56 presents the experimental results from of the handing-over task. As shown in the first four pictures, robot is moving toward the human hand in order to deliver the object with the help of vision system. In pictures 5 and 6, the human will start grasping the object by closing his hand. In pictures 7 and 8, robot will perceive the contact force. It will release the object when the human starts pulling the object. The proposed system will assume that the human as receiver will be the weakest part (blind, elderly etc.) of the task and the robot will play the main role as a positive party to perform the task. In other words, human only needs to close his hand to grasp the object. In conclusion, this part has suggested an automated robot system able to deliver different kinds of objects to the human.

Fig. 5.56 Handing-over object to the human hand

5.4 Conclusion

On the whole, this chapter has shown the importance of using sensor-based robot control with cameras and force/torque sensor systems (visual servoing combined with force control) in three different tasks where both the visual servoing and force control are necessary.

The first scenario illustrates the importance of reducing the impact force during the switching from free space motion to constrained force control by fusing vision and force robot control. The second one is situated in library milieu. The proposed robotic system which integrates vision and force feedback is able to shelve and retrieve imprecisely placed objects according to their alphabetic/numeric codification. Using the proposed automatic decision system, the robot system can decide; which directions should be force controlled and which directions should be vision controlled and how the robot will grasp the target book using two fingers or three fingers, when e.g. the books are stuck together and there are no possibility to enter the index finger and the thumb around the target book. All the developed ideas here could be used in pharmacies, warehouses, factories, supermarkets etc.

The last scenario has to do with handing-over the objects between the human and robot in both directions. The proposed assistant robot system is fully automated. This system is able to deliver and receive model-free objects to the human hand automatically. This work has proposed different real time image processing algorithms in order to detect and to segment loadfree hand, any object carried by human hand and any object located on a flat surface without any information about their model. The proposed control algorithms is able to track the human hand or carried object starting from the free space motion until the full physical human robot integration in real time. The handing-over task is exclusively performed by robot and the human has been considered as the weakest part in this task (elderly, blind or disabled). The fusion of vision and force robot control will ensure the safety of the user during the physical human robot interaction, it will ensure the fulfillment of the grasping/releasing task and it will make the robot able to react to the motion of human hand during the interaction phase.

The proposed system could be implemented in any commercial or industrial robot even if the robot has not the possibility to update its desired position during the motion. Moreover, it could be easily modified in order to fit different scenarios and applications, e.g. in connection with the service, assistance, rescue and mobile robots or even to work simultaneously with the human in industrial human robot teamwork etc.

Chapter 6

Conclusions

As a result of current research and inventions in the advanced robotics field, people are expecting more and more from robots: performance of some complicated tasks either inspired by human action or even needing human robot interaction. However, performing such kinds of tasks requires robotic systems to be more autonomous, user-friendly, dependable and versatile. For this purpose, the scope of this thesis was to increase the autonomy, versatility, dependability and user-friendliness in the particular area of robotics which requires vision/force integration. The contributions of this dissertation are summarized in the following.

6.1 What has been achieved

6.1.1 Autonomy

This work has presented an automatic decision system which decides automatically the appropriate vision/force control structure for different tasks depending on the surrounding environment and the preconditions of tasks, i.e. which subspaces should be vision controlled and which force or position controlled. This work has used all possible types of vision/force control combinations and it has implemented a variety of control structures in different directions in the same task to insure good quality of the control. This strategy has allowed the robot to benefit from all the advantages of different control structures and to perform the complex tasks with no need for re-programming or other human intervention.

6.1.2 User-friendly

This work has designed a high level description of the task, the object and the sensor configuration which is suitable also for inexperienced users. Furthermore, a user interface has been proposed which consists of basic questions which any user can answer before starting the task for entering the prior knowledge and for describing the sensor configuration, the task and the target object easily. Moreover it has assumed a new coordinate system which is convenient to human conception.

6.1.3 Dependability

This work has illustrated how the robot system can depend on its own sensors more than on reprogramming and human intervention. In other words, how the robot system can use all the available information which could be provided by the sensors, especially vision and force

sensors. The integration of vision/force sensors will not only give us information about the target object but will also help us to extract the proprieties of the whole scene, to calculate the graspability, to ensure human safety and to confirm the successful performance of the required task. Furthermore, a new approach of visual servoing has been proposed (4x2D visual servoing) which benefits from the properties of Kinect camera for calculating image integration matrix and the pose of the target object.

6.1.4 Versatility

This work has presented some critical problems in three different applications, where both the visual servoing and force control are necessary and indispensable. Furthermore, it has shown how the integration of vision/force sensors can provide solutions for these problems and how it can improve the robot performance. The proposed approaches are:

Improving impact control: This approach has presented an integrated vision/force robot control for improving the impact control during the switching from free space motion to constrained motion. The case presented in this approach was relatively simple (only in one axis because of the available adjusted controller of the robot system). However, it seems that this approach has potential for improvement in more complicated applications. This concerns not only the quality of contact but also the speed of operation. The speed of operation remained high despite the smoothness and softness of contact force. This section has presented the analysis and experimental testing of vision/force robot control with two kinds of environmental stiffness (medium and hard stiffness). It has shown the advantages of vision feedback in comparison with impact control without visual feedback.

Automated sorting robot system: A new approach to an automated sorting robot system which has integrated vision and force information in order to shelve and retrieve imprecisely placed object according to their alphabetic/numeric codification system. The main contributions of this approach can be summarized as follows:

1. Vision algorithm: an image processing algorithm for detecting different kinds of simple shape objects has been proposed. This algorithm detects objects' position/orientation, characterizes and classifies them and then identifies the codes assigned to objects using SIFT features. The proposed vision algorithm has shown very good performance in detecting the books' boundaries and for recognizing their labels without any limitations; if they are vertical, stuck together, inclined and even if the illumination or viewpoint has been changed. In addition to that, using SIFT features to detect the label has illustrated a better efficiency than previous work which has used the ORC algorithm, for example.

2. Grasping algorithms: two different grasping algorithms have been proposed which are inspired by the human performance of extracting a book from a shelf. In some cases the human uses only two fingers to grasp the book, whereas in other cases he/she uses three fingers to extract the book, especially when the books are stuck together and there are no possibility to enter the index finger and the thumb around the target book. In this way, the proposed robot system was able to grasp books even if the target book had no neighbors, slop

position or it is stuck between two other neighbors. In the first case, the robot system has used only the visual feedback, whereas, in the other case the robot system has integrated vision and force control to grasp the target book.

3. The proposed approach has been implemented in the field of library automation as practical application. It will not only improve the total efficiency of libraries, but it can also relieve the library staff from routine unskilled tasks, thus freeing them to focus on performing other specialized library services. At the same time, the system can eliminate the shelving errors that frequently occur when shelving is performed manually, and valuable time spent looking for books that are misplaced when such errors occur. All the ideas developed here could be used in pharmacies, warehouses, factories or any other place requiring a sorting system.

Automated assistant robot system: A new approach to an assistant robot system for handing-over model-free objects from/to human hands has been proposed. In the proposed system, the transfer object from/to human hand is performed exclusively by robot and the human has been considered as the weakest part in this task (elderly, blind or disabled). The proposed system can tracks the human hand or the carried object starting from the free space motion and ending with the full physical human robot integration in real time. The main contributions of this approach can be summarized as follows:

1. Three vision algorithms related to this approach have been proposed:

- The first one has detected and segmented complicated shape objects on a flat surface without any prior knowledge about their model. The proposed algorithm is able to segment the objects even if they have bad contours, especially when the illumination is constantly changing or even if the objects are located on conveyor or movable surface.
- The two other approaches have detected and segmented any object carried by human hand. (skin color based approach and wrist model based approach). The proposed vision algorithms detected the human hand even if it was closed or it carried objects and segmented the object from the human hand with no idea at all about the model or the color of the object and even when the object had the same color of the human skin. The wrist model based approach can be implemented even if the human wears gloves or has Vitiligo disease. Furthermore, it is able to work in different light conditions starting from complete darkness and ending with different color-temperature lamps. The cycle time of the proposed algorithms is very short ca. 100 ms/frame which gives us opportunity for real time visual tracking.

2. Improving handing-over task. The proposed system has improved the handing-over of objects from human to robot because of the following: The robot system can automatically decide if it is able to grasp the object (graspability), it is able to define the optimal position for grasping the object from the human hand by calculating the tracking point. Furthermore, it has ensured the successful fulfillment of the grasping/releasing task with the help of integrated vision/force control.

3. Real time visual tracking: In this case, the object is carried by human hand and the human will be considered as the weakest part of the operation (negative party). In the proposed system, the robot is able to track the object smoothly and with sufficient speed in real time in spite of the following difficulties: The position of the object is always changing and it can move in all directions, the speed and acceleration of the object motion are not constant (carried by human hand). A commercial robot has been used which means that the desired position and speed cannot be updated or suddenly changed, unless the robot reaches its current target position or until the robot is ready to receive a new command.

4. Improving the interaction phase: The robot system has reacted to any change of an object's position even during the interaction phase with the help of force control. It can help the human to fetch some objects to another place by working as slave during the interaction phase.

5. Safety issue: This work has proposed three safety factors to ensure human safety especially during the interaction phase between human and robot:

- The first two factors are based on visual information (one for the whole human body and the other for the human hand).
- The third safety factor is based on force information.

6.2 An outlook on further work

The future work of the proposed algorithms in this dissertation can be summarized as follows:

6.2.1 Image processing algorithms - Chapter 2

Detection of simple shape object: The proposed algorithm was successfully implemented and tested using a grayscale image (without using any color features or histogram). In the future, it can be combined with some color based image processing algorithms to improve the detecting performance of the boundaries between the neighboring objects. The proposed algorithm in this section could also be optimized in order to reduce the cycle time of detecting the object. The efficiency of the proposed algorithm can be tested by implementing the algorithm to detect other simple objects in various applications, e.g. medicines in pharmacies or storage boxes in warehouses.

Detection of complicated shape objects: The limitation of the proposed algorithms is that the objects should be isolated from each other at least by one centimeter, i.e. the scene should not be cluttered. The proposed algorithm can be improved to detect the objects in cluttered images even if the contours of the objects are not clearly segmented and without any model of them.

Detection of objects carried by human hand: The best possibility to segment any object from the human hand without any idea about the model of the object can be performed by combining both skin color based and wrist model based approaches. However, the present

Kinect technology does not offer this possibility. Future work in this area will go in the direction of integration of both approaches as one approach in a vision device which can deliver color and infrared frames at the same time. Furthermore, the proposed algorithm can be improved in order to segment the object even if the human has rotated his/her hand in different orientations.

6.2.2 Visual servoing approach - Chapter 3

Theoretically, the 4X2D visual servoing approach has been new proposed. However, it still needs more practical experiments to evaluate its efficiency for a variety of objects and different applications.

6.2.3 Automatic decision system – Chapter 4

This dissertation emphasizes the importance of developing the automatic processes of choosing the control structures of the robot. If one looks at the existing research in this field, one can find numerous structures of robot control systems which sometimes confuse the inexperienced user. Hence, future research could concentrate on fusing more types of sensor system, such as force, vision, tactile and other kinds of sensor systems. More basic questions could be investigated and then raised on the inexperienced user in order to let the robot get better overview about the control structure of the proposed task. The user interface of the automatic decision system could be also improved in the future to be based on voice commands.

6.2.4 Practical implementations - Chapter 5

Improving impact control: The case presented in this work was relatively simple (one axis, markers, etc.). As future work, the vision algorithms could be improved in order to eliminate the using of markers. In other words, a vision system could calculate the distance between the end-effector of the robot and the spring-loaded table without any markers or limitations. Furthermore, when the available adjusted controller of robot system has been improved, the proposed algorithms could be tested not only in one axis but also for different axes.

Automated sorting robot system: The proposed system can be developed rapidly using the KINECT camera, which easily detects humans. Hence, the system could work together with humans and the presence of customers will not hinder the work of the robot system and vice versa. Furthermore, the integration of more sensors e.g. vision, force and tactile sensors would improve the system efficiency.

Automated assistant robot system: In future work, other sensors could be integrated with vision/force sensors in order to improve the performance of the task; e.g. tactile sensor can be integrated with the proposed system to optimize the handing-over and grasping tasks. Another sensor is the sensitive skin sensor (Lam, Yip, Qian, & Xu, 2012), (Gandhi & Cervera, 2003) and (Cheung & Lumelsky, 1989) which prevents any collision between any obstacle and the whole robot arm. This sensor would be very necessary, when the human is out of the field-view of

the vision system. This could happen when the robot is not facing the human and then starts rotating to interact with the human. Here, the robot system could use this sensor to ensure the safety of the human by preventing any collision between the human and the whole robot arm. In addition to that, a voice command subsystem could be implemented, so the robot can receive the order of handing-over task by voice commands. In the future, the proposed system could be easily modified to fit various kinds of applications and it could be implemented in service and rescue robots, industrial human robot teamwork etc. especially when this approach has considered the human as the weakest part during the operation (elderly, blind or disabled).

Appendix I

Interaction Matrix of Image Moments

This appendix illustrates the calculation of the general interaction matrix of image moments. Furthermore, the interaction matrix of coordinates of the center of gravity (x_g, y_g), the area (a) and the orientation (θ) of the target object will be calculated.

The general relation of velocity of a point in the world coordinate system $\dot{\vec{p}}$ relative to the camera's velocity could be written as follows:

$$\dot{\vec{p}} = J_p \cdot \vec{v} \tag{I.1}$$

where $\vec{v} = (v_x, v_y, v_z, w_x, w_y, w_z)$ is the velocity of the camera, J_p is the interaction matrix. Consider a camera moving with a body velocity $\vec{v} = (v, w)$ in the world frame and observing a point in the world with relative coordinate to camera $p = (X, Y, Z)$. The velocity of the point relative to the camera frame is:

$$\dot{\vec{p}} = -\vec{w} \times \vec{p} - \vec{v} \tag{I.2}$$

We can write (I.2) in components as:

$$\begin{aligned} \dot{X} &= +Yw_z - Zw_y - v_x \\ \dot{Y} &= +Zw_x - Xw_z - v_y \\ \dot{Z} &= +Xw_y - Yw_x - v_z \end{aligned} \tag{I.3}$$

The perspective projection for coordinates is:

$$x = \frac{X}{Z}, \qquad y = \frac{Y}{Z} \tag{I.4}$$

So,

$$\dot{x} = \frac{\dot{X}Z - \dot{Z}X}{Z^2}, \dot{y} = \frac{\dot{Y}Z - \dot{Z}Y}{Z^2} \tag{I.5}$$

By substituting (I.3), (I.4) in (I.5) we can write the following:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \\ w_x \\ w_y \\ w_z \end{pmatrix} \tag{I.6}$$

For instance, if the object is planar, whose equation expressed in the camera frame is given by:

$$\frac{1}{Z} = Ax + By + C \tag{I.7}$$

And $A = -\Upsilon_1/Z_0$, $B = -\Upsilon_2/Z_0$ and $C = 1/Z_0$, are the 3D parameters. From (I.6) and (I.7), we can write:

$$\begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix} = \begin{pmatrix}-(Ax+By+C) & 0 & (Ax+By+C)x & xy & -1-x^2 & y\\ 0 & -(Ax+By+C) & (Ax+By+C)y & 1+y^2 & -xy & -x\end{pmatrix}\begin{pmatrix}v_x\\ v_y\\ v_z\\ w_x\\ w_y\\ w_z\end{pmatrix} \tag{I.8}$$

so,

$$\begin{aligned}\dot{x} &= -(Ax+By+C)v_x + x(Ax+By+C)v_z + xyw_z - (1+x^2)w_y + yw_z\\ \dot{y} &= -(Ax+By+C)v_y + y(Ax+By+C)v_z + (1+y^2)w_x - xyw_y - xw_z\end{aligned} \tag{I.9}$$

Now we will calculate the Jacobian matrix for the moment feature $J_{m_{ij}}$:

$$\dot{\vec{m}}_{ij} = J_{m_{ij}} \cdot \vec{v} \tag{I.10}$$

As known the equation of the moment can be written as follow:

$$\vec{m}_{ij}(t) = \iint_{R(t)} f(x,y)\,dx\,dy \tag{I.11}$$

where $f(x,y) = x^i y^j$, and $R(t)$ is the part of the image which contains the target object. For determining the analytical form which describes the time variation $\dot{\vec{m}}_{ij}$ of the moments relative to the kinematic screw $\vec{v} = (v, w)$ between the camera and the object, we can write the following:

$$\dot{\vec{m}}_{ij} = \oint_{C(t)} f(x,y)\dot{X}^T n\,dl \tag{I.12}$$

where $C(t)$ is the contour of $R(t)$. By using the Green's theorem (Stewart, 1991), we can write (I.12) in the following form:

$$\dot{\vec{m}}_{ij} = \iint_R \left[\frac{\partial f}{\partial x}\dot{x} + \frac{\partial f}{\partial y}\dot{y} + f(x,y)\left(\frac{\partial \dot{x}}{\partial x} + \frac{\partial \dot{y}}{\partial y}\right)\right] dx\,dy \tag{I.13}$$

Now from (I.9), we deduce the follows:

$$\begin{aligned}\frac{\partial \dot{x}}{\partial x} &= -Av_x + (2Ax+By+C)v_z + yw_x - 2xw_y\\ \frac{\partial \dot{y}}{\partial y} &= -Bv_y + (Ax+2By+C)v_z + 2yw_x - xw_y\end{aligned} \tag{I.14}$$

So

$$\frac{\partial \dot{x}}{\partial x}+\frac{\partial \dot{y}}{\partial y}= -Av_x - Bv_y + (3Ax + 3By + 2C)v_z + 3yw_x - 3xw_y \quad \text{(I.15)}$$

As $f(x,y) = x^i y^j$, we can write:

$$\frac{\partial f}{\partial x} = i\, x^{i-1}\, y^j \qquad \frac{\partial f}{\partial y} = j\, x^i\, y^{j-1} \quad \text{(I.16)}$$

By substituting (I.9), (I.15) and (I.16) in (I.13), we can deduce the interaction matrix for the time variation of the moments:

$$L_{m_{ij}} = \left[m_{v_x}\, m_{v_y}\, m_{v_z}\, m_{w_x}\, m_{w_y}\, m_{w_z}\right] \quad \text{(I.17)}$$

where

$$\begin{cases} m_{v_x} = -i\left(Am_{i,j} + Bm_{i-1,j+1} + Cm_{i-1,j}\right) - Am_{i,j} \\ m_{v_y} = -j\left(Am_{i+1,j-1} + Bm_{i,j} + Cm_{i,j-1}\right) - Bm_{i,j} \\ m_{v_z} = (i+j+3)\left(Am_{i+1,j} + Bm_{i,j+1} + Cm_{i,j}\right) - Cm_{i,j} \\ m_{w_x} = (i+j+3)(m_{i,j+1}) + jm_{i,j-1} \\ m_{w_y} = -(i+j+3)\left(m_{i+1,j}\right) - im_{i-1,j} \\ m_{w_z} = im_{i-1,j+1} - jm_{i+1,j-1} \end{cases}$$

This is the general form of interaction matrix for any particular moment. From the general form in (I.17), we can calculate the form of the zeroth moment ($m_{ij},\ when\ i = j = 0$) which is related to the area $a = m_{00}$ of the object.

$$\begin{aligned} m_{v_x} &= -Aa \\ m_{v_y} &= -Ba \\ m_{v_z} &= a\left[3\left(Ax_g + By_g + C\right) - C\right] \\ m_{w_x} &= 3ay_g \\ m_{w_y} &= -3ax_g \\ m_{w_z} &= 0 \end{aligned} \quad \text{(I.18)}$$

By considering that $1/z_g = \left(Ax_g + By_g + C\right)$, we can deduce the interaction matrix related to the area of the object:

$$L_a = \left[-aA \;\; -aB \;\; a\left(\frac{3}{z_g} - C\right) \;\; 3ay_g \;\; -3ax_g \;\; 0\right] \quad \text{(I.19)}$$

When the object is centered and parallel to the image plane ($A = B = x_g = y_g = 0$), so

$$L_a = [0 \;\; 0 \;\; 2aC \;\; 0 \;\; 0 \;\; 0] \quad \text{(I.20)}$$

Similarly, we can calculate the interaction matrix related to the coordinates ($x_g = m_{10}/m_{00}$) and ($y_g = m_{01}/m_{00}$) of the center of gravity of the object in the image as follows:

$$\begin{aligned}\dot{x}_g &= \frac{m_{00}\dot{m}_{10} - m_{10}\dot{m}_{00}}{m_{00}^2} \\ \dot{y}_g &= \frac{m_{00}\dot{m}_{01} - m_{01}\dot{m}_{00}}{m_{00}^2}\end{aligned} \tag{I.21}$$

By substituting (I.10) in the previous equations, we can write:

$$\begin{aligned}\dot{x}_g &= \left(\frac{m_{00}L_{m_{10}} - m_{10}L_{m_a}}{m_{00}^2}\right)v \\ \dot{y}_g &= \left(\frac{m_{00}L_{m_{01}} - m_{01}L_{m_a}}{m_{00}^2}\right)v\end{aligned} \tag{I.22}$$

Hence, from (I.17), (I.19) and (I.22), we can deduce the interaction matrix related to the center of gravity of the object (x_g, y_g)

$$\begin{aligned}L_{x_g} &= \begin{bmatrix}-1/Z_g & 0 & x_{g_{vz}} & x_{g_{wx}} & x_{g_{wy}} & y_g\end{bmatrix} \\ L_{y_g} &= \begin{bmatrix}0 & -1/Z_g & y_{g_{vz}} & y_{g_{wx}} & y_{g_{wy}} & -x_g\end{bmatrix}\end{aligned} \tag{I.23}$$

where

$$\begin{cases} x_{g_{vz}} = x_g/Z_g + 4(A\eta_{20} + B\eta_{11}) \\ y_{g_{vz}} = y_g/Z_g + 4(A\eta_{11} + B\eta_{02}) \\ x_{g_{wx}} = -y_{g_{wx}} = x_g y_g + 4\eta_{11} \\ x_{g_{wy}} = -\left(1 + x_g^2 + 4\eta_{20}\right) \\ y_{g_{wx}} = 1 + y_g^2 + 4\eta_{02} \end{cases}$$

where η_{11}, η_{02} and η_{20} are the normalized centered moments of order 2, defined as follows:

$$\eta_{ij} = \mu_{ij}/a \ \ with \begin{cases} \mu_{11} = m_{11} - a x_g y_g \\ \mu_{20} = m_{20} - a x_g^2 \\ \mu_{02} = m_{02} - a y_g^2 \end{cases} \tag{I.24}$$

By comparing (I.22) and (I.6), we can deduce that (I.22) is the generalization of the interaction matrix related to the coordinates of a point.

The calculation of the Jacobian matrix of the centered moments will be as follows:

$$\mu_{ij}(t) = \iint_R (x - x_g)^i (y - y_g)^j \, dx \, dy \tag{I.25}$$

By using the Green's theorem as previously, we can write:

$$\begin{aligned}\dot{\mu}_{ij} = \iint_R \Big[& i(x - x_g)^{i-1}(y - y_g)^j(\dot{x} - \dot{x}_g) \\ & + j(x - x_g)^i(y - y_g)^{j-1}(\dot{y} - \dot{y}_g) \\ & + (x - x_g)^i(y - y_g)^j\left(\frac{\partial \dot{x}}{\partial x} + \frac{\partial \dot{y}}{\partial y}\right)\Big] dx \, dy\end{aligned} \tag{I.26}$$

So the general Jacobian matrix of the centered moments could be written as follows:

$$L_{\mu_{ij}} = \left[\mu_{v_x}\ \mu_{v_y}\ \mu_{v_z}\ \mu_{w_x}\ \mu_{w_y}\ \mu_{w_z}\right] \tag{I.27}$$

where

$$\begin{cases} \mu_{v_x} = -(i+1)A\mu_{i,j} - iB\mu_{i-1,j+1} \\ \mu_{v_y} = -jA\mu_{i+1,j-1} - (j+1)B\mu_{i,j} \\ \mu_{v_z} = -A\mu_{wy} + B\mu_{wx} + (i+j+2)C\mu_{i,j} \\ \mu_{w_x} = (i+j+3)\mu_{i,j+1} + ix_g\mu_{i-1,j+1} + (i+2j+3)y_g\mu_{i,j} - 4i\eta_{11}\mu_{i-1,j} - 4j\eta_{02}\mu_{i,j-1} \\ \mu_{w_y} = -(i+j+3)\mu_{i+1,j} - (2i+j+3)x_g\mu_{i,j} - jy_g\mu_{i+1,j-1} + 4i\eta_{20}\mu_{i-1,j} + 4j\eta_{11}\mu_{i,j-1} \\ \mu_{w_z} = i\mu_{i-1,j+1} - j\mu_{i+1,j-1} \end{cases}$$

Hence, the interaction matrix of the 2 order can be calculated as follows:

$$\begin{aligned} L_{\mu_{20}} &= \left[\mu_{20_{vx}} \quad -B\mu_{20} \quad u_{20} \quad t_{20} \quad s_{20} \quad 2\mu_{11}\right] \\ L_{\mu_{02}} &= \left[-A\mu_{20} - B\mu_{02_{vy}} \quad u_{02} \quad t_{02} \quad s_{02} \quad -2\mu_{11}\right] \\ L_{\mu_{11}} &= \left[\mu_{11_{vx}} \quad \mu_{11_{vy}} \quad u_{11} \quad t_{11} \quad s_{11} \quad \mu_{02} - \mu_{20}\right] \end{aligned} \tag{I.28}$$

where,

$$\begin{cases} \mu_{20_{vx}} = -3A\mu_{20} - 2B\mu_{11} \\ \mu_{02_{vy}} = -2A\mu_{11} - 3B\mu_{02} \\ \mu_{11_{vx}} = -2A\mu_{11} - B\mu_{02} \\ \mu_{11_{vy}} = -2B\mu_{11} - A\mu_{20} \\ u_{i,j} = -As_{i,j} + Bt_{i,j} + 4C\mu_{i,j} \\ s_{20} = -7x_g\mu_{20} - 5\mu_{30} \\ t_{20} = 5\left(y_g\mu_{20} + \mu_{21}\right) + 2x_g\mu_{11} \\ s_{02} = -5\left(x_g\mu_{02} + \mu_{12}\right) - 2y_g\mu_{11} \\ t_{02} = 7y_g\mu_{02} + 5\mu_{03} \\ s_{11} = -6x_g\mu_{11} - 5\mu_{21} - y_g\mu_{20} \\ t_{11} = 6y_g\mu_{11} + 5\mu_{21} + x_g\mu_{02} \end{cases}$$

where, the centered moments of the order 3 which are used in the previous equation are:

$$\begin{cases} \mu_{30} = m_{30} - 3x_g m_{20} + 2ax_g^3 \\ \mu_{21} = m_{21} - 2x_g m_{11} - y_g m_{20} + 2ax_g^2 y_g \\ \mu_{12} = m_{12} - 2y_g m_{11} - x_g m_{02} + 2ay_g^2 x_g \\ \mu_{03} = m_{03} - 3y_g m_{02} + 2ay_g^3 \end{cases} \tag{I.29}$$

In the next, we will calculate the interaction matrix related to the object orientation θ which is very interesting feature in our approach. The object orientation θ can be defined as follows:

$$\theta = \frac{1}{2}\arctan\left(\frac{2\mu_{11}}{\mu_{20} - \mu_{02}}\right) \tag{I.30}$$

As $(d\arctan(x))/(dt) = (\dot{x})/(1+x^2)$,

$$\dot{\theta} = \frac{\left(\frac{(\mu_{20}-\mu_{02})\dot{\mu}_{11}-\mu_{11}(\dot{\mu}_{20}-\dot{\mu}_{02})}{(\mu_{20}-\mu_{02})^2}\right)}{1+\left(\frac{4\mu_{11}^2}{(\mu_{20}-\mu_{02})^2}\right)} \quad \text{(I.31)}$$

By rearranging the previous equation we can write:

$$\dot{\theta} = \frac{(\mu_{20}-\mu_{02})\dot{\mu}_{11}-\mu_{11}(\dot{\mu}_{20}-\dot{\mu}_{02})}{(\mu_{20}-\mu_{02})^2+4\mu_{11}^2} \quad \text{(I.32)}$$

So,

$$\dot{\theta} = \frac{(\mu_{20}-\mu_{02})\,L_{\mu_{11}}v-\mu_{11}\left(L_{\mu_{20}}v-L_{\mu_{02}}v\right)}{(\mu_{20}-\mu_{02})^2+4\mu_{11}^2} \quad \text{(I.33)}$$

Hence, the interaction matrix of the object orientation is:

$$L_\theta = \frac{(\mu_{20}-\mu_{02})\,L_{\mu_{11}}-\mu_{11}\left(L_{\mu_{20}}-L_{\mu_{02}}\right)}{(\mu_{20}-\mu_{02})^2+4\mu_{11}^2} \quad \text{(I.34)}$$

By substituting the related equations in the previous one and considering $\Delta = (\mu_{20}-\mu_{02})^2 + 4\mu_{11}^2$, we can deduce the general form of the interaction matrix of the object orientation.

$$L_\theta = \begin{bmatrix}\theta_{vx} & \theta_{vy} & \theta_{vz} & \theta_{wx} & \theta_{wy} & -1\end{bmatrix} \quad \text{(I.35)}$$

where

$$\begin{cases}\theta_{vx} = aA + bB \\ \theta_{vy} = -cA - aB \\ \theta_{vz} = -A\alpha_{wy} + B\alpha_{wx} \\ \theta_{wx} = -bx_g + ay_g + d \\ \theta_{wy} = -ax_g - cy_g + e\end{cases}$$

$$\begin{cases}a = \mu_{11}(\mu_{20}+\mu_{02})/\Delta \\ b = [2\mu_{11}^2 + \mu_{02}(\mu_{02}-\mu_{20})]/\Delta \\ c = [2\mu_{11}^2 + \mu_{20}(\mu_{20}-\mu_{02})]/\Delta \\ d = 5[\mu_{12}(\mu_{20}-\mu_{02}) + \mu_{11}(\mu_{03}-\mu_{21})]/\Delta \\ e = 5[\mu_{21}(\mu_{02}-\mu_{20}) + \mu_{11}(\mu_{30}-\mu_{12})]/\Delta\end{cases}$$

To conclude, this appendix has presented the general interaction matrix of image moments and the interaction matrix of the following features: coordinates of the center of gravity (x_g, y_g), the area (a) and the orientation (θ).

List of Publications

- 1 Journal article
- 1 Workshop paper
- 1 Springer Verlag article
- 12 Conference papers

[1] M. Bdiwi and J. Suchý, “Integration of Vision/force Robot Control Using Automatic Decision System for Performing Different Successive Tasks“, Processing on 8th German Conference on Robotics (Robotik 2014), München, 2014,

[2] M. Bdiwi, A. Kolker and J. Suchý, “Transferring Model-Free Objects between Human Hand and Robot Hand Using Vision/Force Control“, 11th IEEE International Multi-Conference on Systems, Signals & Devices (SSD), Spain, 2014.

[3] M. Bdiwi, A. Kolker, J. Suchý and A. Winkler, “Segmentation of Model-Free Objects Carried by Human Hand: Intended for Human-robot Interaction Applications“, 16th International Conference on Advanced Robotics (ICAR), Uruguay, 2013, ISBN 978-9974-8194-8-1.

[4] M. Bdiwi, A. Kolker, J. Suchý and A. Winkler, “Automated Assistance Robot System for Transferring Model-Free Objects From/To Human Hand Using Vision/Force Control“, In: Guido Herrmann, Martin J. Pearson, Alexander Lenz, Paul Bremner, Adam Spiers und Ute Leonards (Hrsg.) Social Robotics (5th International Conference, ICSR 2013, Bristol, United Kingdom, October 2013, Proceedings), pp. 40-53, Springer, 2013, ISBN 978-3-319-02674-9.

[5] M. Bdiwi, A. Kolker, J. Suchý and A. Winkler, “Real Time Visual and Force Servoing of Human Hand for Physical Human-Robot Interaction: Handing-over Unknown Objects“, Workshop on Human Robot Interaction for Assistance and Industrial Robots in IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, 2013.

[6] M. Bdiwi and J. Suchý, “Storage/Retrieval of Inaccurately Placed Objects Using Robot System and Vision/Force Feedback“, Transactions on Systems, Signals & Devices, TSSD-A, Vol.8, No.1, pp. 1-25, 2013.

[7] A. Kolker, A. Winkler, M. Bdiwi and J. Suchý, “Robot Visual Servoing Using the Example of the Inverted Pendulum“, 10th IEEE International Multi-Conference on Systems, Signals & Devices (SSD), Hammamet, Tunesien, 2013.

[8] M. Bdiwi, J. Suchý and A. Winkler, “Handing-over Model-Free Objects to Human Hand with the Help of Vision/Force Robot Control, 10th IEEE International Multi-Conference on Systems, Signals & Devices (SSD), Hammamet, Tunesien, 2013.

[9] M. Bdiwi and J. Suchý, “Library Automation Using Different Structures of Vision-Force Robot Control and Automatic Decision System“, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Portugal, 2012.

[10] M. Bdiwi und J. Suchý, “Automatic Decision System for the Structure of Vision-Force Robotic Control“, 10th International IFAC Symposium on Robot Control (SYROCO), Dubrovnik, Kroatia, 2012, pp. 172-177.

[11] M. Bdiwi and J. Suchý, “Integrated Vision/Force Robot System for Shelving and Retrieving of Imprecisely Placed Objects“, 7th German Conference on Robotics (Robotik 2012), München, Germany, 2012, pp. 30-35, VDI Verlag, ISBN 978-3-8007-3418-4.

[12] M. Bdiwi and J. Suchý, “Robot Control System with Integrated Vision/Force Feedback for Automated Sorting System“, IEEE International Conference on Technologies for Practical Robot Applications (TePRA), Woburn, MA, USA, 2012.

[13] M. Bdiwi and J. Suchý, “Storage/retrieval of inaccurately placed objects using robot system and vision/force feedback“, 9th IEEE International Multi-Conference on Systems, Signals & Devices (SSD), Chemnitz, Germany, 2012.

[14] M. Bdiwi, J. Suchý and A. Winkler, “Kombinierte Kraftregelung und Visual Servoing eines Roboters mit einem offenen Steuerungssystem“, Scientific Reports, IWKM Moderne Automatisierungstechnik/Robotik, Hochschule Mittweida (FH), Germany, 2011, ISSN 1437-7624.

[15] M. Bdiwi, A. Winkler, J. Suchý and G. Zschocke, “Traded and shared vision-force robot control for improved impact control“, 8th IEEE International Multi-Conference on Systems, Signals & Devices (SSD), Sousse, Tunesien, 2011.

Bibliography

Albitar, C., Graebling, P., & Doignon, C. (2007). " Robust Strucutred Light Coding for 3D Reconstruction". *IEEE 11th International Conference on Computer Vision*, (pp. 1-6). Brazil.

Anderson, R., & Spong, M. (1987). "Hybrid Impedance Control of Robotic Manipulators". *IEEE International Conference on Robotics and Automation*, (pp. 1073-1080).

Baeten, J., & Schutter, J. (2002). "Hybrid vision/force control at corners in planar robotic-contour follwoing". *IEEE/ASME Transactions on Mechatronics*, *7*, pp. 143-151.

Baeten, J., & Schutter, J. (2004). *"Integrated Visual Servoing and Force Control".* Springer, (Springer Tracts in Advanced Robotics 8).

Baeten, J., Bruynicks, H., & Schutter, J. D. (2003). "Integrated Vision/Force Robotic Servoing in the Task Frame Formalism". *The International Journal of Robotic Research*, pp. 941-954.

Baeten, J., Verdonck, W., Bruyninckx, H., & Schutter, J. (2000). "Combining force control and visual servoing for planar contour following". *International Journal of Machine Intelligence & Roboic Control, 2*, 69-75.

Baumgartl, J., & Henrich, D. (2012). "Fast Vision-based Grasp and Delivery Planning for unknown Objects". *7th German Conference on ROBOTIK*, (pp. 98-102).

Bay, H., Ess, A., Tuytelaars, T., & Gool, L. (2008). "Speeded-Up Robust Features". *Computer Vision and Image Understanding, 110*(3), 346-359.

Bergh, M., Carton, D., & Nijs, R. (2011). "Real-Time 3D Hand Gesture Interaction with a Robot for Understanding Directions from Humans". *20th IEEE International Symposium on Robot and Human Interactive Communication*, (pp. 357-362). USA.

Best, J., Molengraft, M., & Steinbuch, M. (2009). "Direct Dynamic Visual Servoing at 1kHz by using the Product as 1.5D Encoder". *IEEE International Conference on Control and Automation*, (pp. 361-366). New-Zealand.

Bicchi, A., & Tonietti, G. (2004). "Fast and Soft Arm Tactics: Dealing with the Safety-Performance Trade-off in Robots Arms Design and Control". *IEEE Robotics and Automation Magazine*, *11*, pp. 22-33.

Bien, Z., Jang, W., & Park, J. (1993). "Characterization and use of Feature-Jacobian matrix for visual servoing". In K. Hashimoto, *In Visual Servoing* (pp. 317-363).

Bischoff, R., & Graefe, V. (2004). "HERMES- a Versatile Personal Assistant Robot". *Proc. IEEE-Speical Issue on Human Interactive Robots for Psychological Enrichment*, *92*, pp. 1759-1779.

Bloblibrary-web. (n.d.). *CvBlobli Wiki*. Retrieved from http://opencv.willowgarage.com/wiki/cvBlobsLib

Bugarin, E., & Kelly, R. (2010). "Robot Vision: Direct Visual Servoing of Planaer Manipulators using Moments of Planar Targets". In A. Ude, *Robot Vision* (pp. 403-428). INTECH.

Cakmak, M., Srinivasa, S., Lee, M., Forlizzi, J., & Kiesler, S. (2011). "Human Preferences for Robot-Human Hand-over Configurations". *IEEE/RSJ International Conference on Intelligent Robots and Systems* , (pp. 1986-1993). USA.

Canny, J. (1986). "A Computational Approach to Edge Detection". *IEEE Transactions on Pattern Analysis and Machine Intelligence, 8*(6), 679-698.

Chaing, H., & Shen, J. (2004). "PID Tuning Rules for Second Order Systems". *5th Asian Control Conference*, *1*, pp. 472-477.

Chaumette, F. (2002). "A first step toward visual servoing using image moments". *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (pp. 378-383).

Chaumette, F. (2004). "Image Moments: A General and Useful Set of Features for Visual Servoing". *IEEE Transactions on Robotics, 20*(4), 713-723.

Chaumette, F., & Hutchinson, S. (2006). "Visual Servo Control Part I: Basic Approaches". *IEEE Robotics & Automation Magazine*, *13*, pp. 82-90.

Chaumette, F., & Hutchinson, S. (2007). "Visual Servo Control Part II: Advanced approaches". *IEEE Robotics & Automation Magazine*, (pp. 109-118).

Chesi, G., & Hashimoto, K. (2002). Static-eye against hand eye visual servoing. *Proceedings of 41st IEEE Conference on Decision and Control* , (pp. 2854-2859). USA.

Cheung, E., & Lumelsky, V. (1989). "Development of Sensitive Skin for a 3d Robot Arm Operating in an Uncertain Environment". *IEEE Conference on Robotics and Automation* .

Chuang, Y., Chen, L., Zhao, G., & Chen, G. (2011). "Hand Posture Recognition and Tracking Based on Bag-of-Words for Human Robot Interaction". *IEEE International Conference on Robotics and Automation*, (pp. 538-543). China.

Collet, A., Berenson, D., Srinivasa, S., & Ferguson, D. (2009). "Object Recognition and Full Pose Registration from a Single Image for Robotic Manipulation". *Proceedings of IEEE international conference on Robotics and Automation*, (pp. 3534-3541).

Collet, A., Martinez, M., & Srinivasa, S. (2011). "The MOPED framework: Object Recognition and Pose Estimation for Manipulation". *The International Journal of Robotics Research*.

Corke, P., & Hutchinson, S. (2000). "Real-Time Vision, Tracking and Contorl". *Proceedings of IEEE International Conference on Robotics and Automation* , (pp. 622-629).

Corke, P., & Hutchinson, S. (2001). "A new partitioned approach to image-based visual servoing control". *IEEE Trans. on Robotics and Automation, 17*(4), 507-515.

Craig, J. (1989). *"Introduction to Robotics Mechanics and Control".* USA: Addison-Wesley Publishing Company.

Cuntoor, N., Collins, R., & Hoogs, A. (2012). "Human-Robot Teamwork using Activity Recognition and Human Instruction". *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (pp. 459-465). Portugal.

Dhanalakshmi, M., & Mamatha, U. (2009). "RFID Based Library Management System". *Proceedings of ASCNT*, (pp. 227-234). India.

Diftler, M., Ambrose, R., Tyree, K., Goza, S., & Huber, E. (2004). "A mobile autonomous humanoid assistant". *4th IEEE/RAS International Conference on Humanoid Robots*, (pp. 133-148).

Duda, R. O. (1973). P. E. Hart, "Pattern Classification and Scene Analysis", New York, Wily.

Duneau, M., & Katz, A. (1985). "Quasiperiodic Patterns". *The American Physical Society*, 2688-2691.

Edsinger, A., & Kemp, C. (2007). "Human-Robot Interaction for Cooperative Manipulation: Handing Objects to One Another". *16th IEEE International Conference on Robot & Human Interactive Communication*, *2*, pp. 1167-1172.

El-laithy, R., Huang, J., & Yeh, M. (2012). "Study on the Use of Microsoft Kinect for Robotics Applications". *IEEE/ION Position Location and Navigation Symposium*, (pp. 1280-1288).

EMVA. (2012). *European machine vision association*. Retrieved from http://www.emva.org/cms/index.php

Epstein, & Zach. (2013). *BGR*. Retrieved from http://bgr.com/2013/02/12/microsoft-xbox-360-sales-2013-325481/

Espiau, B., Chaumette, F., & Rives, P. (1992). "A New Approach to Visaul Servoing in Robotics". *IEEE Transactions on Robotics and Automation, 8*(3), 313-326.

Filliat, D., Battesti, E., Bazeille, S., & Meyer, C. (2012). "RGBD object recognition and visual texture classification for indoor semantic mapping". *IEEE International Conference on Technologies for Practical Robot Applications*, (pp. 127-131). USA.

Finkemeyer, B., Kröger, T., & Wahl, F. M. (2005). "Executing assembly tasks specified by manipulation primitive nets". *VSP and Robotics Society of Japan*, *19*, pp. 591-611.

Finkemeyer, B., Kröger, T., & Wahl, F. M. (2010). "The Adaptive Selection Matrix- A Key Component for Sensor-Based Control of Robotic Manipulators". *IEEE International Conference on Robotics and Automation* , (pp. 3855-3862). USA.

Fischler, M., & Bolles, R. (1981). "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". *Magazine of communication of the ACM*, 381-395.

Flandin, G., Chaumette, F., & Marchand, E. (2000). "Eye-in-hand/ Eye-to-hand Cooperation for Visual Servoing". *IEEE International Conference on Robotics and Automation*, (pp. 2741-2746). USA.

Fofi, D., Sliwa, T., & Voisin, Y. (2004). " A Comparative Survey on Invisible Strucutred Light". *SPIE Electronic Imaging- Machine Vision Applications in Industrial Inspection XII*, (pp. 90-97).

Freedman, B., Shpunt, A., Machline, M., & Arieli, Y. (2010). "Depth Mapping Using Projected Patterns". *United States Patent, pub.No. US 2010/0118123 A1.* USA.

Galambos, C., Matas, J., & Kittler, J. (1999). "Progressive probabilistic Hough transform for line detection". *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, (pp. 554-560).

Gandhi, D., & Cervera, E. (2003). "Sensor Covering of a Robot Arm for collision Avoidance". *IEEE International Conference on Systems, Man and Cybernetics*, *5*, pp. 4951-4955.

Goertz, R. (1954). "Mechanical Master Slave Manipulator". *Nucleonics*, *12*, pp. 45-46.

Goncalves, P., Ferreira, P., & Pinto, P. (2002). "Comparing Visual Servoing Architectures for a Planar Robot ". *Proceedings of the 10th Mediterranean Conference on Control and Automation.* Portugal.

Goron, L., Marton, Z., Lazea, G., & Beetz, M. (2012). "Robustly Segmenting Cylindrical and Box-like Objects in Cluttered Scenes using Depth Cameras". *7th German Conference on ROBOTIK*, (pp. 519-524).

Gourishankar, V., Trybus, L., Rink, R., & Steil, M. (1987). "Controller Design for Robot Arm Guarded and Complaint Motions". *IEEE Transactions on Systems; Man and Cybernetics, 17*(4).

Haddadin, S., Albu-Schäffer, A., & Hirzinger, G. (2007). "Safety Evaluation of Physical Human-Robot Via Crach-Testing". *Robotics: Science and Systems Conference.* USA.

Haddadin, S., Albu-Schäffer, A., Strohmayr, M., Frommberger, M., & Hirzinger, G. (2008). "Injury Evaluation of Human-Robot Impacts". *IEEE International Conference on Robotics and Automation*, (pp. 2203-2204). USA.

Hafez, A., Cervera, E., & Jawahar, C. (2008). "Hybrid Visual Servoing by Boosting IBVS and PBVS". *3rd International Conference on Information and Communication Technologies: From Theory to Applications.*

Hardeman, T. (2008). "Modeling and Identification of Industrial Robots Including Drive and Joint Flexibilities". *PhD thesis, Netherlands Institute for Metal Research.* Netherland.

Hashimoto, K. (2003). "A Review on Vision-Based Control of Robot Manipulators". *Advanced Robotics*, *17*, pp. 969-991. Japan.

Heyer, S., Enjarini, B., Fragkopoulos, C., & Graeser, A. (2012). "Book Detection and Grasping in Library Scenario". *ROBOTIK*, (pp. 389-394). Germany.

Heyer, S., Fragkopoulos, C., Heyer, T., & Gräser, A. (2012). "Reliable Hand Camera based Book Detection and Manipulation in Library Scenario". *13th International Conference on Optimization of Electrical and Electronic Equipment*, (pp. 1486-1492).

Hill, J., & Park, W. (1979). "Real Tiime Control of a Robot with a Mobile Camera". *Proc. 9th SRI International*, (pp. 233-246). United States.

Hocaoglu, M., Bilen, H., Ozgur, E., & Unel, M. (2007). "Model-based vs. Model-free Visual Servoing: A Performance Evaluation in Microsystems". *Proceedings of the 13th IASTED International Conference on Robotics and Applications*, (pp. 316-321). USA.

Hogan, N. (1985). "Impedance Control, An Approach to Manipulation: Part I,II". *Journal of Dynamic Systems Measurement and Control-transactions*, *107.*

Hosoda, K., Igarashi, K., & Asada, M. (1996). "Adaptive hybrid visual servoing/force control in unknown environment". *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, *3*, pp. 1097-1103. Osaka.

Huber, M., Rickert, M., Knoll, A., Brandt, T., & Glasuer, S. (2008). "Human-Robot Interaction in Handing-Over Tasks". *Proceedings of the 17th International Symposium on Robot and Human Interactive Communication*, (pp. 107-112). Germany.

Hussain, A., Siad, Z., Ahamed, N., Sundaraj, K., & Hazry, D. (2011). "Real-Time Robot-Human Interaction by Tracking Hand Movement & Orientation Based on Morphology". *IEEE International Conference on Signal and Image Processing Applications*, (pp. 283-288).

Hutchinson, S., Hager, G., & Corke, P. (1996). "A Tutorial on Visual Servo Control". *IEEE Transactions on Robotics and Automation*, *1*, pp. 651-670.

Hwang, H., & Haddad, R. (1995). "Adaptive Median Filters: New Algorithms and Results". *IEEE Transactions on Image Processing*, *4*, pp. 499-502.

IFR. (2012). *Statistical Department*. Retrieved from http://www.worldrobotics.org/index.php?id=home&news_id=265

JR3-Web. (n.d.). *JR3 Force/torque sensor*. Retrieved from www.jr3.com

Juan, L., & Gwun, O. (2009). "A Comparison of SIFT, PCA-SIFT and SURF". *International Journal of Image Processing, 3*(4), 143-152.

Kase, H., Maru, N., A.Nishikawa, & Yamada, S. (1993). "Visual Servoing of the Manipulator using the Stereo Vision". *Proceedings of International Conference on Industrial Electronics.*

Kemp, C., Edsinger, A., & Torres-Jara, E. (2007). "Challenges for robot manipulation in human environments". *IEEE Robotics & Automation Magazine*, 20-29.

Khalil, W., & Creusot, D. (1989). "SYMORO+: A System for the Symbolic Modelling of Robots". *Proc. 20th Int. Symp. on Industrial Robots*, (pp. 1023-130). Japan.

Khalil, W., Gautier, M., & Lemoine, P. (2007). "Identification of the payload inertial parameters of industrial manipulators". *IEEE International Conference on Robotics and Automation*, (pp. 4943-4948). Italie.

Kim, I., & Inooka, H. (1992). "Hand-over of an object between human and robot". *IEEE International Workshop on Robot and Human Communication*, (pp. 199-203).

Kim, J., Park, J., Hwang, Y., & Lee, M. (2004). "Advanced Grasp Planning for Handover Operation Between Human and Robot: Three Handover Methods in Esteem Etiquettes Using Dual Arms and Hands of Home-Service Robot ". *2nd International Conference on Autonomous Robots and Agents*, (pp. 34-39). New Zealand.

Kim, K., Kwak, K., & Chi, S. (2006). "Gesture analysis for human-robot interaction". *The 8th International Conference on Advanced Communication Technology*, (pp. 1824-1827).

Koscinski, K. (2011). "Hand attractiveness- its determines and association with facial attractiveness". *Behavioral Ecology*, (pp. 1-9).

Kragic, D., & Christensen, H. (2002). *"Survey on Visual Servoing for Manipulation".* Computational Vision and Active Perception Laboratory.

Kröger, T., & Finkemeyer, B. (2011). "Robot Motion Control During Abrupt Switchings Between Manipulation Primitives". *Workshop on Mobile Manipulation at the IEEE International Conference on Robotics and Automation.*

Kröger, T., & Wahl, F. M. (2010). "Online Trajectory Generation: Basic Concepts for Instantaneous Reactions to Unforeseen Events". *IEEE Transactions and Robotics, 26*, 94-111.

Kröger, T., & Wahl, M. (2008). "Multi-Sensor Integration and Sensor Fusion in Industrial Manipulation: Hybrid Switched Control, Trajectory Generation, and Software Development". *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, (pp. 411-418). Korea.

Lam, T., Yip, H., Qian, H., & Xu, Y. (2012). "Collision Avoidance of Industrial Robot Arms using an Invisible Sensitive Skin". *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (pp. 4542-4543).

Lambrecht, J., & Krüger, J. (2012). "Spatial Programming for Industrial Robots based on Gestures and Augmented Reality". *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (pp. 446-472). Portugal.

Lange, F., Suppa, M., & Hirzinger, G. (2012). "Control with a Complaint Force-Torque Sensor". *ROBOTIK*, (pp. 490-495). Germany.

Lee, E. (1999). "Force and Impact Control for Robot Manipulators Using Time Delay". *Proceeding of IEEE International Symposium on Industrial Electronic* , *1*, pp. 151-156. USA.

Lee, E., Park, J., Schrader, C., & Chang, P. (2003). "Impact When Robots Act Wisely". *Proceedings of IEEE International conference on Robotics & Automation*, (pp. 3692-3697). Taiwan.

Lopez-Damian, E., Sidobre, D., Tour, S. d., & Alami, R. (2006). "Grasp Planning for Interactive Object Manipulation". *5th International Symposium on Robotics and Automation*, (pp. 1-6). Mexico.

Lowe, D. (2004). "Distinctive Image Features from Scale-Invariant Key-points". *Internatinoal Journal of Computer Vision*, *60.*

Main, A., Bennamoun, M., & Owens, R. (2006). "Three-Dimensional Model-Based Object Recognition and Segmentation in Cluttered Scenes". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*, pp. 1584-1601.

Mainprice, J., Sisbot, E., Simeon, T., & Alami, R. (2010). "Planning Safe and Legible Hand-over Motions for Human-Robot Interaction". *IARP Workshop on Tech. Challenges for Dependable Robot in Human Environments.*

Malis, E. (2002). "A Unified Approach to Model-based and Model-free Visual Servoing". *European Conference on Computer Vision*, *4*, pp. 433-447. Denmark.

Malis, E. (2002). "Survey of vision-based robot control". *European Naval Ship Design, Captian Computer IV Forum*, (pp. 1-16). France.

Malis, E., Chaumette, F., & Boudet, S. (1999). "2 1/2 D Visual Servoing". *IEEE International Transactions on Robotics and Automation*, *15*, pp. 238-250.

Mangin, O., & Oudeyer, P. (2012). "Learning to recognize parallel combinations of human motion primitives with linguistic descriptions using non-negative matrix factorization". *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (pp. 3268-3275). Portugal.

Maples, J., & Becker, J. (1986). "Experiments in force control of robotic manipulators". *In Proc. IEEE Conference Robot and Automation*, (pp. 695-702).

Mason, A., & MaxKenzie, C. (2005). "Grip forces when passing an object to a partner". *Experimental Brain Research, 163*, 173-187.

Mason, T. (1981). "Compliance and Force Control for Computer Controlled Manipulators". *IEEE Transactions on Systems, Man and Cybernetics* , (pp. 418-432).

McCormick, W., & Schwartz, H. (1993). "An Investigation of Impedance Control for Robot Manipulators". *Int. Journal of Robotics Research, 12*(5), 473-489.

Morano, R., Ozturk, C., Conn, R., & S.Dubin. (1998). "Structured Light Using Pseudorandom Codes". *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 322-327.

Morel, G., Malis, E., & Boudet, S. (1998). "Impedance Based Combination of Visual and Force Control". *Proc. IEEE International Conference on Robotics and Automation*, (pp. 1743-1748).

Mutto, C. (2012). *"Time-of-Flight Cameras and Microsoft Kinect".* SpringerBriefs in Electrical and Computer Engineering.

Nagata, K., Oosaki, Y., Kakiura, M., & Tsukune, H. (1998). "Delivery by Hand between Human and Robot Based on Fingertip Force-Torque Information". *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (pp. 750-757).

Nakano, Y., Y.Kihara, Sakimoto, K., & Hayashi, Y. (2003). "Automated Library System with Retrieving and Respositing Robot". *United States Patend, pub.No. US 6,535,790 B2.* USA.

Nelson, B., Morrow, D., & Khosla, P. (1996). "Robotic Manipulation Using High Bandwith Force and Vision Feedback". *Mathematical and Computer Modeling*, *24*, pp. 11-29.

Nelson, B., Morrow, J., & Khosla, P. (1995). "Improved Force Control Through Visual Servoing". *Proceeding of the American Control Conference*, (pp. 380-386). USA.

Neumann, R. (1991). *"Rechnergestützte Implementierung digitaler Regler".* Ludwigsburg, Germany: AGT Verlag.

Neumann, R. (1991). "Rechnergestützte Implementierung digitaler Regler". *AGT Dokumentation Verlag*.

Nieuwenhuisen, M., Stückler, J., Berner, A., Klein, R., & Behnke, S. (2012). "Shape-Primitive Based Object Recognition and Grasping". *7th German Conference on ROBOTIK*, (pp. 468-472). Germany.

Nixon, M., & Aguado, A. (2008). "Feature Extraction and Image processing". *Academic Press.*

OpenNi-Web. (n.d.). *Standard framework for 3D sensing*. Retrieved from http://www.openni.org/reference-guide/

Osypiuk, R., Kröger, T., Finkemeyer, B., & Wahl, F. (2006). "A Two-Loop Implicit Force/Position Control Structure, Based on a Simple Linear Model: Theory and Experiment". *Proceedings of IEEE International Conference on Robotics and Automation*, (pp. 2232-2237). Florida.

Pan, Y., Ge, S., He, H., & Chen, L. (2009). "Real-time face detection for human robot interaction". *18th IEEE International Symposium on Robot and Human Interactive Communication*, (pp. 1016-1021).

Pari, L., Sebastian, J., Angel, L., & Rueda, J. (2009). "Image Base Visual Servoing: Estimation of the image Jacobain by using lines in a stereo vision system". *IEEE International Conference on E-Learning in Industrial Electronics*, (pp. 109-114).

Parker, J., & Paul, F. (1988). "Method And Apparatus For Controlling Impact Force During Rapid Robotic Acquisition of Object". *United States Patent, pub.No. 4,783,107.* USA.

Phung, S., Bouzerdoum, A., & Chai, D. (2005). "Skin segmentation using color pixel: analysis and comparison". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (pp. 148-154).

Prats, M., Sanz, P., & Pobil, A. (2005). "Model-based Tracking and Hybrid Force/Vision Control for the UJI Librarian Robot". *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (pp. 1090-1095).

Prewitt, J. (1970). "Object enhancement and extraction", In Picture Processing and Psychopictorics, New York.

Primesense-web. (n.d.). *NITE Middleware*. Retrieved from http://www.primesense.com/solutions/nite-middleware/

Prokop, R., & Reeves, A. (1992). "A survey of moment-based techniques for unoccluded object representation and recognition". *Proc. Computer Vision, Graphics, Image Processing Conf.*, *54*, pp. 438-460.

Raibert, M., & Craig, J. (1981). "Hybrid Position/Force Control of Manipulators". *Journal of Dynamic Systems, Measurement and Control*, 126-133.

Rajinder Sodhi, Brett Jones. (2010). *Kinect-Projector Calibration.* Urbana-Champaign: University of Illinois.

Ramos-Garijo, R., Prats, M., Sanz, P., & Pobil, A. D. (2003). "An Autonomous Assistant Robot For Book Manipulation in a Library". *IEEE International Conference on Systems, Man and Cybermetics*, (pp. 3912-3917).

Ranko, Z., Angel, V., Pedro, G., & Angel, L. (2006). "An Architecture for Robot Force and Impact Control". *IEEE Conference on Robotics, Automation and Mechatronics*, (pp. 1-6). Bangkok.

Reinhart, G., Zaidan, S., & Hubele, J. (2010). "Is Force Monitoring in Cooperating Industrial Robots Necessary?". *6th German Conference on Robotik*, (pp. 523-530). Germany.

Salisbury, K. (1980). "Active Stiffness Control of a Manipulator in Cartesian Coordinates". *19th IEEE Conference on Decision and Control*, (pp. 95-100).

Salvi, J., Pags, J., & Battle, J. (2004). "Pattern Codification Strategies in Structured Light Systems". *Pattern Recognition*, (pp. 827-849).

SCHUNK-Web. (n.d.). *SCHUNK Company*. Retrieved from http://www.schunk-modular-robotics.com

Seraji, H. (1994). "Adaptive Admittance Control: An Approach to Explicit Force Control in Complaint Motion". *IEEE Int. Conference on Robotics and Automation*, (pp. 2705-2712).

Shirai, Y., & Inoue, H. (1973). "Guiding a Robot by Visual Feedback in Asembling Tasks". *Pattern Recognition Pergamon Press*, *5*, pp. 99-108. Great Britian.

Shupnt, A., Tikva, P., Zalesvsky, Z., & Ha'ayin, R. (2008). "Depth- Varying Light Fields For Three Dimensional Sensing". *United States Patent, pub.No. US 2008/0106746 A1.* USA.

Sian, N., Sakaguchi, T., & Yokoi, K. (2006). "Operating Humanoid Robots in Human Environments". *Proceedings of the Robotics, Science & Systems Workshop on Manipulation for Human Environments*, (pp. 1-6). Philedelphia.

Smisek, J., Jancosek, M., & Pajdla, M. (2011). "3D with Kinect". *IEEE International Conference on Computer Vision Workshops*, (pp. 1154-1160).

Spiller, A., & Verl, A. (2012). "Force Controlled handling with Cooperating Industrial Robots". *ROBOTIK*, (pp. 496-501). Germany.

Stampfer, D., Lutz, M., & Schlegel, C. (2012). "Information Driven Sensor Placement for Robust Active Object Recognition based on Multiple Views". *IEEE International Conference on Technologies for Practical Robot Applications* , (pp. 133-138). USA.

Stäubli. (1992). *"AdeptVision Version 13.0".* Adept Technology.

Stefanescu, D. (2011). "Strain Gauges and Wheatstone Bridges- Basic Instrumentation and New Applications For Electrical Measurement of Non-Electrical Quantities". *Systems, Signals and Devices (SSD), 2011 8th International Multi-Conference on*, (pp. 1-5). Tunis.

Stewart, J. (1991). *"Calculus"* (6th ed. ed.). Pacific Grove, CA: Brooks/Cole.

Suchý, J. (2013). "Visual servoing". Chemnitz University of Technology, Germany.

Suthakorn, J., Lee, S., Zhou, Y., Thomas, R., & Choudhury, S. (2002). "A Robotic Library System for an Off-Site Shelving Facility". *IEEE International Conference on Robotics and Automation*, (pp. 3589-3594).

Timothy, C., & D.Plutt. (2003). "Automated Storage Library with Multiple Independent Robots". *United States Patent, pub.No. US 6,570,734 B2.* USA.

Tombari, F., & Stefano, L. D. (2010). "Object Recognition in 3D Scenes with Occlusions and Clutter by Hough Voting". *Fourth Pacific-Rim Symposium on IMage and Video Technology*, (pp. 349-355).

Tomizawa, T., Ohya, A., & Yuta, S. (2003). "Object Posture Recognition for Remote Book Browsing Robot System". *IEEE/RSJ International Conference on Intelligent Robots and Systems*, *3*, pp. 3218-3223.

Tomizawa, T., Ohya, A., & Yuta, S. (2003). "Remote Book Browsing System using a Mobile Manipulator". *IEEE International Conference on Robotics and Automation*, (pp. 256-261).

Viager, M. (2011). *Analysis of Kinect for mobile robots.* Technical University of Denmark.

Volpe, R., & Khosla, P. (1993). "A Theoretical and Experimental Investigation of Explicit Force Control Strategies for Manipulators". *IEEE Transactions and Automatic Control*, *38*, pp. 1634-1650.

Waiboer, R. (2007). "Dynamic Modeling Identification and Simulation of Industrial Robots -For Offline Programming of Robotized Laser Welding". *PhD thesis, Netherlands Institute for Metal Research.* Netherland.

Wang, J., Zhang, G., Zhang, H., & Fuhlbrigge, T. (2008). "Force Control Technologies For New Robotics Applications". *Technologies for Practical Robot Applications*, (pp. 143-149). USA.

Weiss, L., Sanderson, A., & Neuman, C. (1985). "Dynamic Visual Servo Control of Robots: An adaptive image-based approach". *Proceedings of IEEE International Conference on Robotics and Automation*, (pp. 662-668).

Weiss, L., Sanderson, A., & Neuman, C. (1987). "Dynamic Sensor Based Control of Robots with Visual Feedback". *IEEE Journal of Robotics and Automation*(5), 404-417.

Weng, S., & Young, K. (1996). "Robot Impact Control Inspired By Human Reflex". *International Conference on Robotics and Automation*, (pp. 2579-2585). Minnesota.

Whitney, D. (1985). "Historical perspective and state of the art in robot force control". *Proceedings of IEEE International Conference on Robotics and Automation*, *2*, pp. 262-268.

Wiaboer, R., & Aarts, R. (2005). "Modelling and Identification of a Six Axes Industrial Robot". *ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, (pp. 1-10). USA.

Winkler, A., & Suchý, J. (2005). "Novel Joint Space Force Guidance Algorithm with Laboratory Robot System". *16th IFAC World Congress.* Prague.

Winkler, A., & Suchý, J. (2006). "Sensorless Force Guided Motions of an Industrial Robot Based on Motor Currents". *37th International Symposium on Robotics and 4th German Conference on Robotics.* München.

Xiao, Q., Hu, X., Gao, S., & Wang, H. (2010). "Object Detection Based on Contour Learning and Template Matching". *Proceedings of the 8th World Congress on Intelligent Control and Automation*, (pp. 6361-6365). China.

Yakimovsky, Y., & Cunningham, R. (1978). "A system for extracting three-dimensional measurements from a stereo pair of TV cameras". *Computer Graphics and Image Processing*, (pp. 195-210).

Yaun, M., Farbiz, F., Mason, C., & Yin, T. (2009). "Robust Hand Tracking Using a Simple Color Classification Technique". *The International Journal of Virtual Reality, 8*(2), 7-12.

Yin, X., & Zhu, X. (2006). "Hand Posture Recognition in Gesture-Based Human-Robot Interaction". *1St IEEE Conference on Industrial Electronics and Applications*, (pp. 1-6).

Yoshie, J. (1997). "Book storage/retrieval apparatus". *United States Patent, pub.No US 5,690,463.* USA.

Zeng, G., & Hemami, A. (1997). "An Overview of Robot Force Control". *Journal Robotica*, *15*, pp. 473-482.

Zinn, M., Khatib, O., & Roth, B. (2004). "A New Actuation Approach for Human Friendly Robot Design". *Int. J. of Robotics Research, 23*, 379-398.

List of Figures

List of Tables

Translations

TECHNISCHE UNIVERSITÄT CHEMNITZ

Entwicklung und Erprobung von Algorithmen zur Kombinierung der Vision- und Kraftregelung mit automatischem Entscheidungssystem

Von der Fakultät für Elektrotechnik und Informationstechnik
der Technischen Universität Chemnitz

genehmigte

Dissertation

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

vorgelegt von

Dipl.-Ing. Mohamad Bdiwi
geboren am 25. August 1985 in Aleppo, Syrien

eingereicht am

20.01.2014

Gutachter:

Prof. Dr.-Ing. Jozef Suchý
Prof. Dr.-Ing. Michael Gerke

Tag der Verleihung:

Zusammenfassung

In Übereinstimmung mit den aktuellen Forschungen und Erfindungen im Bereich der fortgeschritten Robotik, erwarten die Menschen mehr und mehr von einem Roboter. Er soll komplizierte Aufgaben erfüllen, die entweder inspiriert vom menschlichen Verhalten sind oder die die Mensch-Roboter Interaktion voraussetzen. Allerdings stellen dieser Art von Aufgaben besondere Anforderungen an das Robotersystem. Es muss autonom, benutzerfreundlich, zuverlässig und vielseitig sein. Ziel dieser Arbeit ist nun die Erhöhung der Autonomie, der Benutzerfreundlichkeit, der Zuverlässigkeit und der Vielseitigkeit des Robotersystems in dem Bereich wo es erforderlich ist Vision- und Kraftregelung zu kombinieren.

1. Beiträge und Herausforderungen

Die Kerninhalte dieser Arbeit werden im Folgenden geordnet und zusammengefasst, damit es wird gezeigt, wie Sie die Autonomie, die Benutzerfreundlichkeit, die Zuverlässigkeit und die Vielseitigkeit des Robotersystems erhöhen können.

1.1 Autonom

In dieser Arbeit wird ein automatisches Entscheidungs-System präsentiert, das die entsprechende Vision/Kraft Reglerstruktur für unterschiedliche Aufgaben automatisch auswählen kann, abhängig von der Umgebung, den Sensoren und den Anforderung zum Lösen der Aufgaben. Dies bedeutet, dass das System automatisch entscheiden kann, welche der kartesischen Richtungen (Unterräume) sollten per Vision geregelt werden und in welchen sollte die Kraft oder die Position geregelt werden. Zum Erfüllen einer Aufgabe kann das Entscheidungs-System aus allen möglichen Kombinationen von Vision- Kraft- und Positionsregelungen auswählen. Außerdem ist es möglich den verschiedenen kartesischen Richtungen unterschiedliche Reglerstrukturen zuzuweisen, um eine gute Qualität der Regelung zu gewährleisten. Diese Strategie erlaubt es dem Roboter von den Vorteilen der verschiedenen Vision/Kraft/Position Reglerstrukturen zu profitieren und so komplexe Aufgaben zu erfüllen, ohne jedes Mal durch den Nutzer neu programmiert werden zu müssen.

1.2 Benutzerfreundlichkeit

In dieser Arbeit wird eine Beschreibungsmethode für die Aufgabe des Roboters, für das zu manipulierende Objekt und für die Sensorkonfiguration vorgeschlagen, die auf einer höheren Abstraktionsebene liegt und so für unerfahrene Anwender geeignet ist. Darüber hinaus wird für das System eine Benutzerschnittstelle vorgeschlagen. Sie besteht aus grundlegenden Fragen, die jeder Benutzer beantworten kann, um die Sensorkonfiguration, die Aufgabe und das Zielobjekt leicht zu beschreiben. Außerdem wird ein neues, auf den Menschen ausgelegtes Koordinatensystem präsentiert.

1.3 Zuverlässigkeit

In diese Arbeit wird gezeigt, wie das Verhalten des Robotersystems, statt durch Umprogrammierung und Intervenieren durch den Menschen, durch seinen Sensoren

verändert werden kann. Mit anderen Worten, das Roboter System kann alle verfügbaren Informationen verwenden, die von den Sensoren, insbesondere von Vision- und Kraftsensoren zur Verfügung gestellt werden. Die Integration von Vision- und Kraftsensoren wird uns nicht nur Informationen über das Zielobjekt liefern, sondern sie wird uns auch helfen Merkmale der gesamten Szene zu extrahieren, um z. B. die „Greiffähigkeit/Greifbarkeit“ zu berechnen, die Sicherheit des Menschen zu gewährleisten und die gestellte Aufgabe erfolgreich zu bewältigen. Darüber hinaus wurde ein neuer Ansatz des Visual Servoing vorgeschlagen (4x2D Visual Servoing), der mit den Möglichkeiten der Kinect-Kamera die Orientierung des Zielobjektes in der Bildebene berechnet.

1.4 Vielseitigkeit

In dieser Arbeit werden einige kritische Probleme in drei verschiedenen Anwendungen aufgezeigt, bei denen sowohl das Visual Servoing und die Kraftregelung notwendig und unverzichtbar sind. Dabei wird gezeigt, wie durch die Kombination von Kamera und Kraftsensor diese Probleme gelöst werden können und somit die Leistungsfähigkeit des Robotersystems erhöht werden kann. Die vorgeschlagenen Ansätze sind im Einzelnen:

Verbesserung der Stoßkraftregelung: Mit diesem Ansatz wird eine kombinierte Vision/Kraft Roboterregelung präsentiert, mit dem Ziel eine unbekannte Umgebung mit möglichst geringen Kraftspitzen zu kontaktieren. In dieser Arbeit wurde dieser Algorithmus, aufgrund des zur Verfügung stehenden Robotersystems, nur für einen kartesischen Freiheitsgrad realisiert. Jedoch kann dieser Ansatz jederzeit auf mehrere Freiheitsgrade erweitert werden, um auch bei komplizierten Aufgaben seine Anwendung zu finden. Dabei ist nicht nur die Regelgüte beim Kontaktieren der Umgebung wichtig, sondern auch die Verfahrgeschwindigkeit des Roboters im freien Raum. Diese sollte trotz geringer Kraftspitzen beim Kontaktieren möglichst hoch sein. Der vorgeschlagenen Algorithmus wurde an einer variablen Kontaktumgebung (mittlere und harte Steifigkeit) experimentell verifiziert.

Automatisches Roboter-Sortier System: Hier wird ein neuer Ansatz eines automatisierten Sortierroboter System präsentiert, der die Bild- und die Kraftinformation kombiniert, um ungenau platzierte Objekte nach ihrer alphabetischen/numerischen Kodierung zu sortieren. Die wichtigsten Beiträge dieses Ansatzes, der z. B. im Szenario einer automatisierten Bibliothek vorkommen kann, lassen sich wie folgt zusammenfassen:

1. Vision Algorithmus: Hier wird ein Bildbearbeitungsalgorithmus vorgeschlagen, um verschiedene einfach geformte Objekte zu detektieren. Dieser Algorithmus erkennt die Position und die Orientierung der Objekte, charakterisiert und klassifiziert sie. Danach identifiziert er die zugewiesenen Kodierungen der Objekte mit der Hilfe von SIFT Merkmalen. Die vorgeschlagene Vision Algorithmus zeigte eine sehr gute Leistungsfähigkeit, z. B. um Bücher ohne Einschränkungen zu greifen, egal ob sie einzeln-vertikal positioniert sind, zusammen geschoben sind oder auch schräg orientiert sind. Auch bei sich ändernder Beleuchtung und Blickrichtung der Kamera arbeitete der Algorithmus erfolgreich.

2. Algorithmen zum Greifen: Hier werden zwei verschiedene Greifalgorithmen vorgeschlagen, die von der menschlichen Erfahrung bei Herausnehmen eines Buches aus einem Regal

inspiriert sind. Dabei verwendet der Mensch in einigen Fällen nur zwei Finger, um das Buch zu fassen, während er in anderen Fällen mit drei Fingern arbeitet, insbesondere wenn die Bücher eng zusammen angeordnet sind und es keine Möglichkeit gibt mit dem Zeigefinger und dem Daumen zwischen die Bücher zu gelangen. Im ersten Fall verwendet das Robotersystem nur die Rückführung der Kamerainformation. Im anderen Fall kombiniert das Robotersystem Vision- und Kraftregelung, um das gewünschte Buch zu greifen. Auf diese Weise kann das vorgeschlagene Robotersystem Bücher in jeder Situation greifen.

Die beiden vorgeschlagenen Ansätze wurden, wie schon kurz erwähnt, im Bereich der Bibliotheksautomatisierung praktisch implementiert. So könnte nicht nur die Gesamteffizienz der Bibliotheken verbessert werden, sondern die Bibliotheksmitarbeiter auch von Routineaufgaben entlastet werden. Parallel dazu kann das System Fehler in den Bücherregalen beseitigen, die häufig bei der manuelle Durchführung auftreten können. Der hier entwickelte Ansatz könnte auch auf andere Anwendungen übertragen werden, so z. B. auf Apotheken, Lagerhäuser, Fabriken usw.

Automatisches Assistenz-Robotersystem: In diesem Abschnitt wird ein neuer Ansatz für ein Assistenz-Robotersystem vorgeschlagen, das Objekt an den Menschen übergeben oder von ihm übernehmen kann. Dabei sind keine Modelle der Objekte erforderlich. Die Transferaufgabe soll ausschließlich durch den Roboter durchgeführt werden, da der Mensch mit geringer Aktivität angesetzt wird, z. B. ältere, blinde oder behinderte Menschen. Das vorgeschlagene System kann die leere Hand des Menschen oder ein gegriffenes Objekt in Echtzeit verfolgen, beginnend von der Bewegung des Objektes im freien Raum und bis hin zum vollständigen physischen Kontakt zwischen Mensch und Roboter. Die wichtigsten Beiträge dieses Ansatzes lassen sich wie folgt zusammenfassen:

1. Zunächst werden drei Bildverarbeitungsalgorithmen vorgeschlagen:

- Der erste Algorithmus erkennt kompliziert geformte Objekte, die z. B. auf einem Tisch angeordnet sind, ohne Vorkenntnisse über ihre Geometrie zu haben, und segmentiert sie. Die Segmentierung erfolgt auch dann erfolgreich, wenn die Konturen des Objektes undeutlich sind, die Beleuchtungssituation sich verändert oder auch wenn die Objekte beweglich sind, wie z. B. auf einem Förderband.
- Die beiden anderen Ansätze erkennen Objekte die durch den Menschen gegriffen wurden und nun von ihm gehalten werden. Die Segmentierung erfolgt dabei zum einen über die Hautfarbe und zum anderen über den sog. Handgelenk-Modell Ansatz. Mit den hier vorgeschlagenen Bildverarbeitungsalgorithmen konnte die Hand des Menschen erfolgreich segmentiert werden, unabhängig von der aktuellen Fingerstellung und auch unabhängig davon ob eine Objekt gegriffen wurde oder nicht. Dabei ist es bei dem Ansatz über die Hautfarbe nicht notwendig eine exakte Information über die Hautfarbe des jeweiligen Menschen zu haben, da diese aus dem vorher detektierten Gesicht extrahiert wird. Ebenso ist keine Information über das gegriffene Objekt notwendig. Im Gegensatz zum „Hautfarbe-Ansatz“ funktioniert der Handgelenk-Modell Ansatz auch wenn der Mensch Handschuhe trägt oder die Vitiligo

Krankheit hat. Des Weiteren arbeitet er auch unter schlechten Lichtverhältnissen und auch in völliger Dunkelheit. Die Zykluszeit der vorgeschlagenen Algorithmen war dabei sehr klein (ca. 100 ms pro Bild), was es ermöglicht visuelle Echtzeitverfolgung zu realisieren.

2. Verbesserung der Objektübergabe: Im hier vorgeschlagenen System konnte der Prozess der Objektübergabe zwischen Mensch und Roboter aus folgendem Grund verbessert werden: Das Robotersystem kann automatisch entscheiden, ob der Roboter das Objekt greifen kann (Greifbarkeit). Außerdem kann es die optimale Greifposition am Objekt bestimmen. Darüber hinaus kann das Robotersystem durch die kombinierte Vision/Kraftregelung entscheiden, zu welchem Zeitpunkt es sinnvoll ist das Objekt zu greifen oder es loszulassen.

3. Echtzeit Verfolgung per Vision: In diesem Szenario wird das Objekt von Menschenhand geführt und der Mensch wird als das schwächste Glied im Prozess angesehen. In dem hier vorgeschlagenen System kann der Roboter das Zielobjekt problemlos und mit ausreichender Geschwindigkeit in Echtzeit verfolgen, trotz der folgenden Herausforderungen: (1) Die Position des Objekts wird immer verändert und es kann sich in alle Richtungen bewegen. (2) Die Geschwindigkeit und die Beschleunigung des Objekts sind ebenfalls nicht konstant, da die Bewegung von der menschlichen Hand durchgeführt wird. (3) Außerdem wurde ein kommerzielles Robotersystem verwendet, was bedeutet, dass man die gewünschte Bewegung des Endeffektors nur zu bestimmten Zeitpunkten mit Hilfe von Kommandos für Gelenk- oder Linearinterpolation vorgeben kann. Ein direkter Zugriff zu den Sollwerten der Lageregelkreise oder gar zu Geschwindigkeitssollwerten oder Stromsollwerten auf Gelenkebene besteht nicht.

4. Verbesserung der Interaktionsphase: Das Robotersystem reagiert auf jede Änderung der Position des Objekts während der Interaktionsphase mit der Hilfe einer Nullkraftregelung (Handführen). Diese Funktionalität kann dem Menschen während der Interaktionsphase helfen, ein Objekt mit Hilfe des Roboters an einen anderen Ort zu transportieren.

5. Sicherheitsaspekt: In diese Arbeit werden drei Sicherheitsfunktionalitäten vorgeschlagen, um die Sicherheit des Menschen während seiner Interaktion mit dem Roboter zu gewährleisten:

- Zwei Funktionalitäten basieren auf der Kamerainformation (eine für den ganzen menschlichen Körper und die andere für die menschliche Hand).
- Die dritte Sicherheitsfunktionalität nutzt die Messwerte des Kraft-/ Momentsensors.

Die vorliegende Dissertation gliedert sich insgesamt in sechs Kapitel. Nach einer Einführung und dem Blick auf vorangegangene Arbeiten werden im Kapitel 2 die vorgeschlagenen Bildverarbeitungs-algorithmen in allen ihren Einzelheiten beschrieben. Im Kapitel 3 wird der neu entwickeltet 4x2D Visual-Servoing Ansatz ausführlich vorgestellt. Danach wird das automatische Entscheidungssystem und seine Schnittstellen im Kapitel 4 präsentiert. Das Kapitel 5 beschreibt die vorgeschlagenen praktischen Anwendungen und zeigt die gewonnenen Versuchsergebnisse auf. Die Arbeit schließt mit der Zusammenfassung und eine Ausblick auf zukünftige Arbeiten.

Inhaltverzeichnis

Versicherung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direct order indirect übernommenen Gedanken sind als solche kenntlich gemacht.

Weitere Personen waren an der Abfassung der vorliegenden Arbeit nicht beteiligt. Die Hilfe eines Promotionsberaters habe ich nicht in Anspruch genommen. Weiter Personen haben von mir keine geldwerten Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt.

Chemnitz, den 20. Januar 2014

Mohamad Bdiwi

Curriculum Vitae

Personal Information

Name	Mohamad Bdiwi
Date of Birth	25th August 1985
Place of Birth	Aleppo, Syria
Martial Status	Married (one daughter)

Education

Oct. 2008 – present	PhD student in Chemnitz University of Technology, department of robotic system.
2002-2007	Dipl.-Ing. (Honors degree, 1st rank with average 83.11%) Department of automation control and industrial automation, University of Aleppo, Syria.
1999-2002	General secondary school, Aleppo, Syria with average 90.7%.

Professional experience

Aug. 2013 – Present	Developing engineer in Fraunhofer Institute IWU, Chemnitz
2007 – 2008	Lecturer and research assistant, Faculty of Electrical and Electronic Engineering, University of Aleppo, Syria.
Mar. 2007 – Sep. 2008	Project engineer in RTA automation company, Aleppo, Syria.

Award

I have been awarded ALBASEL award five times for five successive years by minister of high education in Syria, because I have gotten 1st rank in all my study years.

Refereed Publications

1 Journal article	1 Workshop paper
1 Springer Verlag article	12 Conference papers