The UJI Librarian Robot

Mario Prats, Ester Martínez-Martín, Pedro J. Sanz and Angel P. del Pobil

{mprats, emartine, sanzp, pobil}@icc.uji.es

Robotic Intelligence Laboratory, Universitat Jaume I, Castellón, Spain

Communicating autor: Mario Prats mprats@uji.es, +34-964-387048

Abstract

This paper describes the UJI Librarian Robot, a mobile manipulator that is able to autonomously locate a book in an ordinary library, and grasp it from a bookshelf, by using eye-in-hand stereo vision and force sensing. The robot is only provided with the book code, a library map and some knowledge about its logical structure and takes advantage of the spatio-temporal constraints and regularities of the environment by applying disparate techniques such as stereo vision, visual tracking, probabilistic matching, motion estimation, multisensor-based grasping, visual servoing and hybrid control, in such a way that it exhibits a robust and dependable performance. The system has been tested, and experimental results show how it is able to robustly locate and grasp a book in a reasonable time without human intervention.

Keywords*: mobile manipulator, multisensor-based manipulator and grasping, hybrid control*

1. Introduction

Nowadays, there is a great effort in robotics research to make robots capable of working in everyday scenarios, co-existing with humans. Currently, mobile service robots can be found in relatively unstructured environments such as museums, hospitals, etc. They typically make use of the available state-of-the-art technology to combine sensor-based navigation, localization and mapping, as well as obstacle avoidance capabilities and advanced user interfaces. However, service applications that include sensor-based manipulation in human environments are still very scarce. We believe that libraries offer a perfect start point to develop such applications, since they are human scenarios that are both ordinary and semi-structured. In this paper, we present the *UJI Librarian Robot*, a system that not only is able of navigating in an ordinary library searching for a book, but it can autonomously extract it from the bookshelf with a robot arm, and bring it to the user.

The UJI librarian robot is a complete robotic system designed to assist users in the library at Universitat Jaume I (UJI). The user requests a book by using a modification of the standard web-based library interface, and the robot is then in charge of locating the book in the library, retrieve it and carry it to an assigned position. The system is only provided with the book code, which appears written on a label on each book spine. Using a vision system, it finds the particular bookcase and bookshelf where the book is, and integrates vision and force sensing to locate the target book and grasp it with an ad-hoc parallel-jaw gripper.

One of the main challenges of our work is to integrate disparate techniques ranging from visual perception or multisensor-based grasping to visual servoing, hybrid control or navigation, in such a way that not only each subsystem works correctly, but also the system as a whole exhibits a robust and dependable performance.

Related previous work at Tsukuba University [1] included a teleoperated system aiming also at providing remote book browsing. In the authors' opinion, two important tasks needed improvement: optimization of the manipulation planning and book recognition; these topics are addressed in our research and satisfactorily solved.

To the best of our knowledge, there is only one active project in the world that follows a goal similar to ours; namely, the *Comprehensive Access to Printed Materials* (CAPM) project [2]. Its main objective, presently under progress, will be to enable real-time browsing of printed materials through a web interface. An autonomous mobile system has been developed to retrieve items from bookshelves and carry them to scanning stations located in an off-site facility. For this to be possible, the books have to be stored in cases with a special design.

In our work, the environment is not modified for making things easier: books are not placed into special boxes for easier retrieval, we try to grasp the books as they can be found in current libraries, without making strong assumptions. Instead of adapting the world to the robot, we rather try to adapt the robots to the world and take advantage of the opportunities that the constraints of the particular environment offer. Moreover, we use a visual system for the robot to search autonomously for the book as well as force sensing for the final grasp execution. We believe that force feedback is a must in robotic manipulation as a way of coping with uncertainty and vision errors.

This paper is organized as follows: section 2 describes general hardware and software aspects of the robot. Section 3 describes the environment where the robot evolves and presents in detail the procedures followed by the robot to locate the book in the library. Section 4 deals with the vision algorithms that are used for locating and recognizing particular books. Section 5 introduces some visual and force control techniques used to guide the gripper towards the desired book and to grasp it. Finally, experimental results are described in section 6.

2. System description

2.1. Hardware

An image of the UJI Librarian Robot can be seen in Figure 1. It shows a mobile manipulator composed of:

- Mobile robot. The base of the librarian robot is the Nomad XR4000 (Nomadic Technologies). It is equipped with radio Ethernet, two rings with 24 sonar and 24 infrared sensors each, bumpers and a SICK laser rangefinder for navigation and collision avoidance. It has been modified to integrate the arm controller inside.

- Robot arm. A Mitsubishi PA-10 manipulator is mounted on the mobile robot. This manipulator is a lightweight robot arm with seven degrees of freedom. The redundant degree of freedom can be used to avoid collisions at the same time that the task is being performed. At this moment, kinematic redundancy is dealt with by using the manufacturer's low-level robot library; this issue is not described in this article.

Figure 1

The robot arm is endowed with a VidereDesign stereo pair, a parallel-jaw gripper, a JR3 force/torque sensor and infrared distance sensors. The stereo cameras are mounted in an eye-in-hand configuration. The six degree-of-freedom force sensor is mounted in the wrist between the robot end-effector and the gripper. Finally, some infrared distance sensors are attached along the arm structure, they can be used to detect obstacles and avoid collisions. The whole system is controlled autonomously from the computer inside the mobile robot which is in charge of controlling the mobile robot itself, the manipulator and force sensor through a different PCI card, the gripper via an expansion input/output board, and the camera using the IEEE1394 interface.

2.2. Software

The system is able to search and retrieve a book requested by a user. The operation starts when the user requests a book. The robot is then in charge of locating the book in an ordinary library, extract it and bring it to the user. The only initial information is the book code, written on a label which is *read* by the vision system. This general application integrates several inter-disciplinary skills. For performing these tasks, several software modules have been implemented. The global behaviour of the system emerges from the cooperation between the following modules:

- User interfaces. The first one simply transfers the code of the desired book to the robot, it is a modification of the standard web-based library interface. The second one can be used by an operator, or the user himself, for monitoring the robot activity. It can serve for supervisory or shared control if necessary (Figure 2); for instance to validate a book label which was not recognized with sufficient confidence. This interface is not described in this paper; for details, refer to [3].

- Book localization. It uses a library representation in order to decide which bookcase contains the desired book. It also includes algorithms for assisting the search for a book within the bookcase. By using a map of the library, it controls the mobile robot so that it reaches a given position.
- Vision module. It takes images from the cameras and processes them in order to locate book labels and to extract the text through optical character recognition.
- Book retrieval. It is in charge of controlling the robotic arm and gripper for safely retrieving the book.

Figure 2

3. Book localization

The task environment is the UJI Library in Castellón (Spain). In this context, we define a bookcase as a homogeneous furniture structure which is composed of three modules. Each module contains seven shelves (see Figure 3). We have also defined a suitable data structure for storing a model of the library, in which the physical layout is mapped to the logical structure. Although the model has been defined with particular information of our library, it can be easily adapted to any other library by means of the following user-configurable parameters:

- The number of buildings
- The number of floors in each building
- The number of bookcases in each floor, with their respective location and structure, i.e., number of modules, and number of shelves, and sizes thereof.

Initially, only the code of the first book in a bookcase is stored manually, though additional information about books in that bookcase is added automatically later during the normal functioning of the system. Using this representation, once the system is provided with a book label, it can determine the bookcase that will probably contain it, together with the location of this bookcase inside the library. The above parameters should be properly adapted to the configuration of a different library, but the search algorithm would remain the same.

Figure 3

We assume that books are classified using the *U.S. Library of Congress Classification* (LCC), which assigns a unique code (also called *signature*) to each

book. This code is composed of two fields: the class number and the book number. The system could easily be adapted to another classification scheme.

Once the system has received the code of the book to find, the first problem is to locate its position in the library. This is done in three steps:

- Finding the bookcase that contains the book
- Navigation to that bookcase
- Finding the book within the bookcase

3.1. Finding the bookcase

The robot is provided with a map of the library that is used for navigation, as shown in Figure 4. Each bookcase object in our logical representation stores the code of the first book in it. The code of the target book is compared with those codes until the target bookcase is found. Our logical bookcase objects also include the position inside the library map. Thus, given a book, the robot is able to locate the position in the library map of the bookcase that contains it. This is done quickly, just by looking up in its library representation, prior to any robot motion.

Figure 4

3.2. Navigation

Once the robot knows the position of the bookcase that should contain the book, the map of the library is used to plan a path and take the mobile platform from its current location to the desired one (Figure 4). Navigation is achieved using the Player/Stage software, by using sonar, infrared and laser range sensors that provide collision avoidance capabilities. It is not the purpose of this article to describe the navigation module since it is based on state-of-the-art techniques for mapping and localization that we are not describing here. They take advantage of the holonomic nature of the Nomad XR4000 and are based on previous work [4, 5, 6, 7] with further improvements based on Bayesian methods for probabilistic localization and occupancy grid mapping [8, 9].

When the robot arrives at the target position, it is approximately aligned with respect to the bookcase so that the camera vision plane is somewhat parallel to it. This condition is further accomplished by using vision for computing the book plane and aligning the camera with it, as we explain later on in section 4.1.

3.3. Finding the book

Using the available library representation, the robot is able to locate the bookcase that should contain the book, but not the exact position of the book inside it. For finding it, a visually-guided search along the shelves and modules of the bookcase is performed. Instead of searching the book one by one in the entire bookcase, an intelligent search algorithm has been implemented that takes advantage of the fact that books are arranged following a particular sequence. This algorithm works as follows (see flowchart in Fig. 5):

1. First, it is checked whether the bookcase is the correct one. For this, an image of the books at the left of the uppermost shelf containing books is taken, as shown in Figure 3a (remember the vertical position of the shelves is known). Book labels can be extracted from an image (see next section) so that it is easy to find the leftmost label, i.e. the one having other labels on its right but none on its left. If the code of the first *few* books in the shelf are greater than the desired one, then the target book must be in a previous bookcase. The system will move to the previous bookcase and will start again. Otherwise, if the first labels are lower than the required one, the book might be found in this bookcase, and the next step of the algorithm is executed.
2. At this point, the robot arm is positioned to be able to capture an image of containing the labels of the first *few* books placed at the lowest shelf. If their codes are greater than the target, then the requested book is in this module and the next step is executed. But if they are lower, the book could be either on this lowest shelf or on the next module (Figure 3b). Hence, the robot tries to locate the book in this shelf, by reading all the codes. If it is not found and all the codes are lower than the required one, the robot will move to the bottom shelf of next module and will repeat this step. During this process, if the robot finds a few books with codes greater than the target, it concludes that the book is not placed at the right position.
3. Once the module where the book should be has been identified, the correct shelf must be determined. From the bottom shelf, the arm moves vertically and reads the first book code in each shelf (Figure 3c). When it finds a code that is lower than the requested one, the target shelf has been found, and the final step is executed.

4. Finally, a linear search is performed on the shelf until the book is found. If, before finding the book, the shelf end is reached or a few code numbers greater than the target are found, it is concluded that the book is not placed at the right position.

Note that, in some situations, the book might not be found due to the fact that a user could have put it in a slightly wrong place. The system reacts to this situation by describing a small, rectangular area, centered in the predicted position. The book is then searched within this range. Similarly, one of the codes read during the search process may correspond to books incorrectly placed, this is the reason why decisions are only taken after reading a *few* codes (a configurable parameter).

Figure 5

4. Vision module

The algorithm for finding a book within a bookcase, makes use of vision for locating, reading and comparing book codes. In this section we show the methods that are currently being used for reading book codes from images. This problem involves image segmentation, tracking of book labels and optical character recognition (OCR).

The vision module must be able to read book codes from single images, but also from sequences of images. At some situations the search algorithm takes an image of the leftmost part of a shelf. This is a single image from which book codes must be read. In this case, only image segmentation and OCR is needed. But in the final stage of the book search, a continuous movement of the camera along the direction of the bookcase is performed. This produces a sequence of images. Here, book tracking is also necessary, in order to keep a list of which labels have already been processed by the OCR module, and which have not.

We use a probabilistic tracking method in order to match labels through time, exploiting the spatio-temporal constraints of the problem. Using vision, book labels are segmented and located in the image. Next, they are sent to the OCR module in order to read the identification codes. From the readings, the search algorithm decides where to move the manipulator so that the book is found as soon as possible. The purpose of the tracking is to keep all labels located at any time, so that if the OCR module fails to recognize one of them reliably, it can be tried again later, from another point of view.

Note that at some situations, just one image is enough to conclude that the book we are searching is not on that shelf, and to decide to move to another very different position. So, although we are interested in a tracking algorithm, it is a requirement that the whole algorithm works also on single images. We propose a sequential and modular method in order to achieve these tasks, composed of four steps: label segmentation, probabilistic matching, motion estimation, and probabilistic selection.

The following constraints and assumptions about the problem are adopted:

1. We assume that the camera is adequately placed in front of the shelf, so that the camera plane is approximately parallel to the book plane. This can be accomplished by estimating the book plane by means of vision, as it will be explained in section 4.1.
2. All labels have white background and black foreground. In addition, they contain four lines of text (Figure 6).
3. Labels are all placed more or less at the same height (Figure 6).
4. Labels cannot overlap. This is not an assumption but a property of the problem that can be used to better detect regions.
5. Labels are rectangular. The vertical size is approximately the same for all. The width depends on the width of the book.
6. All the characters on the label are visible on the book spine, i.e. the book is not too thin.
7. The environment is static. Changes on the image are only due to camera movements. Labels can only appear or disappear at the sides of the image, when the velocity is non zero.

It is worth noting that all these hypothesis are realistic and in agreement with our particular library environment.

Figure 6

4.1. Plane estimation

When the mobile manipulator arrives at a position in front of the bookcase, the eye-in-hand camera must be aligned with the book plane, in order to have an optimal view of the books and make the image processing easier. As robot localization is not enough to ensure that this ideal position has been reached, we

use vision for computing the book plane and automatically aligning the camera with it.

This step is currently done open-loop, by estimating the book plane once, at the initial camera position. It could also be performed in closed-loop by following a visual servoing approach. However, in this case we would need to select a set of stable visual features and to be able to track them in the image at each iteration. Precision in positioning would be better, but the execution would not be as reliable as open-loop. Even if making use of robust features and robust trackers, it would be very difficult to track the whole set of features during the alignment motion, even more knowing the reflective nature of most book surfaces. There is also the possibility to extract a whole set of new features at each iteration and computing the correspondences on the stereo image. However, this approach would be computationally expensive. In both cases, robot velocity should be small during the process, thus increasing the book delivery time. Open-loop precision was experimentally shown to be enough for this particular application. Thus, the minimalist solution was adopted, avoiding tracking and allowing fast alignment motion.

We use the well-known pin-hole camera model and epipolar geometry for linking the information contained in 2D images with the 3D structure of the projected world [10].

As the librarian robot has an eye-in-hand stereo camera, two different views are taken automatically. Both lenses point in the same direction, but are separated by a baseline of 90mm. We take the camera frame, *C*, at the midpoint between the lenses, and another frame is attached to each lens, *I* and *D*, as shown in Figure 7.

Figure 7

Thus, taking *C* as the world center, the 3D coordinates of the reconstructed points will be defined with respect to this system. By the transformations ${}^{I}T_{C}$ and ${}^{D}T_{C}$, both views are also defined in the common frame *C*. Thus, both projection matrices can be computed with the following expressions [10]:

$$\begin{aligned} P &= K^{i} P_{I}\, {}^{I}T_{C} \\ P' &= K^{d} P_{I}\, {}^{D}T_{C} \end{aligned} \qquad (1)$$

Where K^i and K^d are the calibration matrices for each camera. Knowing $P=[P_{3\times3} \quad p_{3\times1}]$ and $P'=[P'_{3\times3} \quad p'_{3\times1}]$, the fundamental matrix can be easily computed by using the following well-known result from epipolar geometry [10]:

$$F=[p'-P'P^{-1}p]_x P'P^{-1} \qquad (2)$$

Once F is known, correspondences can be found in a fast and robust way [10] by looking for SIFT features in one of the images, and applying the sum of squared differences (SSD) method along the epipolar line in the other image. Having the correspondences, the 3D structure of the world can be computed. Figure 8a shows two images taken by the stereo cameras and a set of nine correspondences and three epipolar lines. Due to the particular arrangement of the cameras, both views point in parallel directions and the epipolar lines are horizontal. As books have normally different colors and text on their sides, the images are well-textured and it is easy to find many correspondences. In order to fit a plane, at least three 3D points must be reconstructed. The plane is computed by using standard least squares minimization. The greater the number of correspondences, the better the adjustment will be. Figure 8b shows the plane that is obtained from the correspondences. In the current implementation, outliers are not taken into account, making the plane estimation quite sensible to errors in the feature matching process. The use of robust estimators is one of the aspects that will be addressed in future work. The book plane could also be estimated by following a dense matching approach, and trying to fit a plane to a cloud of 3D points, but this method would take more computational time.

Figure 8

Having the plane equation in the camera frame, *C*, and knowing the relationship between the camera and the end-effector frame, ${}^{E}T_{C}$, an open-loop end-effector displacement can be easily computed so that the camera is placed perpendicular to the plane, as it is desired for vision and grasping algorithms to work properly.

4.2. Segmentation

The first step aims at locating the labels inside the image. For this, we propose a local segmentation algorithm with automatic threshold detection. For a $N\times M$ greyscale image I, we first divide it into L vertical intervals of $C=M/L$ columns each one. We define the luminosity of the row i of interval k as:

$$l_K(i) = \frac{\sum_{j=kC}^{(k+1)C} I_{ij}}{C} \qquad (3)$$

that is, the mean intensity of the *ith* row in interval k, being I_{ij} the value of the pixel with column j and row i (in the range $[0,255]$). Then, we find the row with more luminosity for each interval k:

$$L_K = \max\{\ l_k(i)\ \}\ \ \forall i \in [0, N-1] \qquad (4)$$

$$LR_K = \arg\max\{\ l_k(i)\ \}\ \ \forall i \in [0, N-1] \qquad (5)$$

From assumptions 2 and 3, it is easy to see that horizontal lines in rows LR_K will frequently intersect with book labels. We say frequently, because for an image interval where all the books are white, the row with more luminosity LR_K could not pass through the label. However, this is not a major problem because, since book and label are of the same color, we can treat it as a large label and refine the detection in later stages.

Merging this property with equations (4) and (5), we can say that L_K corresponds to the mean intensity of labels. In Figure 6 we show the parameters involved in the segmentation process.

The mean intensity of the row with more luminosity, L_K, can be used for segmentation purposes. We know that a book label is the lightest region we can find in the image, and its color value is around L_K. We can segment locally each interval k using as threshold the value of $(3/4)L_K$, chosen experimentally. The result will be a binary image where labels are visible, along with some regions of the same color. Of course, the quality of the segmentation depends on the total number of intervals. For just one interval taking all the image, there is one L_0 that is the mean color of all the labels in the image. However, there are some labels that are darker than others, because they are older or due to shadows. In this case, L_0 would not be a good approximation of label intensity for all of them, and segmentation would not be precise (note that this would be the equivalent to a global segmentation). This problem is solved by taking more intervals into account and by segmenting the image locally. Of course, the labels of two books are more probable to have a similar intensity than those of six books. Then, higher values for L give a better segmentation, at the expense of lower execution speed.

Also note that the automatic selection of the threshold makes the segmentation algorithm robust to illumination conditions and illumination changes. Concretely, robustness in global illumination is due to the assumption that labels are the lightest regions in the image. The algorithm automatically detects the lightest rows and its mean values, which are used to estimate a correct segmentation threshold. Moreover, local illumination changes are detected thanks to the use of intervals which allow to segment separately each part of the image.

We have said that the rows LR_K intersect with all the labels, and thus they can be used to get the boundaries. The process is as follows:

For each interval $k \in [0, L-1]$:

a. Consider the row with more luminosity, LR_K, and find in the segmented image the next white pixel in that row that does not belong to any label, i.e., it has not been assigned previously to any label.
b. That pixel belongs to a new label, and we can use a standard contour extraction algorithm to get the boundary.
c. Repeat a and b until the end of the interval

Finally, a curvature analysis on the boundaries is done in order to detect angles and lines. Due to assumption 5, detected boundaries that do not contain two pairs of parallel lines are rejected. Figure 6 shows the final result of the segmentation algorithm for a real image.

4.3. Probabilistic matching and motion estimation

If we want to extend our system to work over sequences of images, we can apply the segmentation process to each of them. But if we also want to follow the movement of a certain book, then we need a tracking method. In this application, tracking is especially interesting because we want to keep a list of the books that have already been successfully processed by the OCR module. So, if we apply OCR to all the books of an image, and then the camera moves a little bit to the right so that a new book appears, then we must pass the new label to the OCR module, and not the already processed ones.

We propose a model-based probabilistic approach [11, 12] for matching labels in two consecutive images. Rather than using the location of the labels in the first image to predict its location in the second image and avoid a whole segmentation,

we take advantage of the good performance of the segmentation algorithm and execute a whole segmentation each time. With this, we aim to improve the localization of the labels following a probabilistic framework. For example, segmentation is not always perfect, and it is possible to start with a low quality image of the label. Then, we apply segmentation to all images while the camera is moving. If the label is segmented correctly in one image, our probabilistic algorithm will remember its features for helping future segmentations.

The matching algorithm tries to link books detected in the current image with those detected in the previous one. For this, a good representation of a label is needed. We describe a label by its four vertices, represented as points A, B, C, D in the two dimensional image space. Points are ordered so that A corresponds to the top-left corner, B to the top-right, C to the bottom-right and D to the bottom-left. We can describe a label as a four dimensional vector $S_i = (A_i, B_i, C_i, D_i)$.

Following with the notation, labels segmented at a given time t, are represented by the vector $S^t = (S_1, S_2, ..., S_n)$, where n is the total number of books in the image. From this, the algorithm tries to match elements of S^t with elements in S^{t-1}.

The matching is done following a probabilistic approach. Given two labels, S_i and S_j, we define a metric for measuring the distance between them as:

$$M(S_i, S_j) = \frac{d_{A_iA_j} + d_{B_iB_j} + d_{C_iC_j} + d_{D_iD_j}}{4} \qquad (6)$$

where d_{xy} is the point-to-point distance between points x and y. We also define the random variable X= “distance between two corresponding labels in two consecutive images”', and then:

$$X \rightarrow N(v, \sigma^2) \qquad (7)$$

i.e., X is distributed by a Normal with mean v and variance σ^2. The mean v is an estimation of the image velocity and the variance is an estimation of the error in the velocity.

Of course, the distance between the same label in two consecutive images depends on the image velocity v, that can be estimated either by a calibrated

camera with known motion, or by computing the image flow, as we do. The variance in the distribution model can be chosen experimentally, but it will not affect the tracking algorithm.

Our system works by computing the distance of all possible matches of labels between S^t and S^{t-1}. Then, we apply the probability model in (7) in order to find the most probable pairs. Three cases can occur:

1) A label in S^t does not match with any label in S^{t-1}

   This can happen in two situations. The first one is at the start of the algorithm, when S^{t-1} has no elements. The second one is when a new label appears in the image, due to camera motion. In both cases, we assign an internal identification number to the book, that will be used for tracking.

2) A label in S^{t-1} does not match with any label in S^t

   This means that the label has disappeared. As we have an estimation of the image flow, we can apply some restrictions about the place where labels can disappear. For example, for a static image ($v = 0$), no labels can disappear. However, if we displace the camera to the right, then labels can only disappear through the left side of the image.

   Our system takes into account these constraints. If a label disappears where it is not possible to, it is created again in S^t and displaced according to the prediction of the velocity. In this way, we can deal with occlusions or with errors in the segmentation process. The motion of lost labels can be predicted from the motion of those which are matched.

3) A label $S_i \in S^t$ is matched with a label $S_j \in S^{t-1}$

   That means that S_i in the current image is the same that S_j in the last image, so we have successfully matched the label. But once more, segmentation problems can arise and it is possible to have a bad extraction in S_i, while S_j is good. To cope with this problem, we define two additional random variables: H= “height of a label” and W= “width of a label”. These variables are also distributed according to a Normal distribution, with a mean and a variance that can be chosen experimentally.

When two labels match, the probability models are computed on S_i and S_j. If S_i has a very low probability of being a good label, and S_j has high probability, then we overwrite S_i with the good one and displace it in the image by using the image flow estimation. With this we ensure that if a label is well detected at a given time, future problems with it will be solved by using the good prediction.

But in most of the cases, S_i and S_j will be good, and then the match will be used to update the image flow to be used in future predictions. The image flow is a 2D vector f computed as follows:

$$f = \frac{t_A + t_B + t_C + t_D}{4} \qquad (8)$$

with:

$$t_A = A^t - A^{t-1} \qquad (9)$$

$$t_B = B^t - B^{t-1} \qquad (10)$$

$$t_C = C^t - C^{t-1} \qquad (11)$$

$$t_D = D^t - D^{t-1} \qquad (12)$$

The velocity used in the probability distribution is defined as the module of the image flow (average for all label pairs), $v = |f|$.

4.4. Probabilistic selection

The last step in the vision module is the probability-based selection. The probability model of a label is used here to eliminate those labels with a low probability of being amenable for grasping. With this, we can discriminate books that are too wide for being grasped by the robot, for example.

Moreover, this stage is in charge of selecting those books that must be passed to the OCR module. In this way, only new labels are selected, i.e. those in the current image which have not been found or have failed to be recognized in the previous one.

4.5. OCR Module

The recognition module takes as input a set of labels and it first tries to locate text lines inside them. For each label, we take parallel lines, perpendicular to the segment that joins points A and D, and compute the luminosity for each line in a way similar to that of section 4.2 The maxima of this function represent the background while the minima represent text lines. In this way, the four lines of text are located.

After that, the angle of the label is computed as:

$$\alpha = \arctan \frac{A_y - B_y}{B_x - A_x} \qquad (13)$$

A rotation of $-\alpha$ is applied to the label before it is passed to the OCR, in order to transform tilted labels into vertical ones. The utility of locating the separation between text lines is that each line can be passed separately to the OCR, and then we can apply some restrictions about the characters that can appear in each line. For example, we know that the first line of text should only contain letters, while the fourth line is a year, i.e., only numbers. If we process the lines separately, we can use these constraints for the OCR to reduce errors.

As OCR core, we use the GOCR software [13] that is being developed under the GNU Public License. This software is able to extract the text included in images. It works with all kind of fonts, and can be trained for better performance in particular applications.

5. Book retrieval

When the desired book is found in the image, the grasping process starts. An ad-hoc parallel jaw gripper is used to accomplish the task of extracting a book from a bookshelf (Figure 9). It is endowed with special purpose fingertips. Some initial hypothesis must be assumed for the grasping to be feasible:

- Books must be graspable by our robot arm. So, its physical properties (i.e. length, width, weight, etc.) will be always suitable to the available manipulation capabilities (i.e. geometry of the gripper, etc.).
- The spines of the books from the robot point of view, oriented towards the outside the shelf, are all of them, approximately, in the same spatial plane.

- Moreover, it is assumed that the books are not pressed together on the shelf in such a way as to impede the insertion of the gripper fingertips.
- We assume that the manipulator is adequately placed so that the camera plane is approximately parallel to the book plane. The camera is placed at a distance from the books suitable for the best vision performance (*d* in Figure 9). This is achieved by the techniques already explained in section 4.1.

Figure 9

We try to grasp the book from the front. As it can be seen in Figure 9, the gripper fingertips are quite thin and fit well into the separation between books. Instead of inserting both fingers at the same time, which could be a difficult task, the left *nail* is inserted first, and, next, the right one. For allowing this operation, the left fingertip has been made longer than the right one. It must be noted that the task is not to insert the fingers in a rigid surface. Books are not rigidly attached to the shelf, and many of them are flexible. This allows to deal with the small positioning errors coming from the open-loop alignment of section 4.1. The grasping process follows these steps:

- Reaching the book by means of a force/vision control law
- Looking for the left side of the book and inserting the left fingertip, until the right one hits with the book.
- Looking for the right side of the book and inserting the right fingertip while the left one remains in its previous location.
- Taking the book out of the bookshelf

5.1. Reaching

For accomplishing the first task, we use a closed-loop scheme based on visual and force information, under the Task Frame Formalism (TFF), that gives us a robust, reliable and fast method. The TFF, as defined in [14, 15], is a powerful concept for the specification and compliant execution of manipulation tasks [16, 17]. The space of possible task directions is split into two different spaces: directions which are constrained by the environment and directions which are not.

For directions which are not constrained by the environment, the robot can be velocity-controlled, without taking into account forces in this direction, because no obstacles will appear. However, for constrained directions, the robot must be force-controlled, in order to avoid breaking anything. Many tasks can be

described with the TFF [18]. The classical example is the table cleaning task, where there is a constrained direction which is perpendicular to the table plane, whereas the parallel directions are free.

Thus, a hybrid control law [17, 19, 20, 21] has been implemented, combining force and visual servoing (see Figure 10) and enabling the control of two degrees of freedom: one in the perpendicular direction to the books, and the other one in the parallel direction. For the parallel direction we employ image-based visual servoing [22, 23] by using the edges of the book as visual features. The perpendicular direction is force-guided in order to make contact with the book. Once contact is made, the control law is switched to force sensing only, as explained in [24]. If the book is tilted, one more degree of freedom is used for gripper orientation.

Figure 10

For the vision control law we use only one visual feature: the left edge of the book to grasp. The edge is located in the image by using the vision subsystem presented in section 4, and it is represented as a segment (linking points A and D). Then, the segment is tracked while the end effector moves to the convergence state, guided by a task function. In practice, we use the central point of the segment as visual feature, so the feature vector becomes:

$$s = (E_X) \qquad (14)$$

where E_X is the x component of the central point of the segment $\overrightarrow{AD}$, $E = \frac{A-D}{2}$. The desired state s^* is preprogrammed. It has been obtained by moving the gripper to the desired position and storing the position in the image where the visual feature remains. We can set the image error to be:

$$e = \hat{L}^{+}\left(s - s^*\right) \quad (15)$$

In our particular case, as the visual primitive vector has only one dimension, the interaction matrix can be set to the identity. Imposing an exponential decrease of the error, we have:

$$\dot{e} = -\lambda \cdot e \qquad (16)$$

Merging **(13)** and **(14)** we obtain:

$$\tau_c = -\lambda \cdot \hat{L}^{+} \cdot \left(s - s^{*}\right) \qquad (17)$$

where τ_c is the kinematic vector expressed in the camera coordinate system and L is the interaction matrix. τ_c can be transformed into another kinematic vector τ_e, expressed in the end effector coordinate system by a kinematic transformation matrix, in the form:

$$\tau_e = {}^{e}\mathrm{T}_c \cdot \tau_c \qquad (18)$$

where ${}^{e}\mathrm{T}_c$ is the screw transformation matrix between camera and effector frame. This control law lets us control the lateral motion of the end effector that aligns the gripper with the book. In order to control the perpendicular direction, we define a simple force control law in a similar way. A desired force f^{*} is set by learning in an off-line stage. The error with the current force f leads to a kinematic vector with a direction perpendicular to the books.

The coupling between both control laws lets the gripper approach the books, while performing a lateral motion that keeps the gripper aligned with the desired book. The other degrees of freedom are not controlled. The coupled control law finishes when both the visual servoing and the force control have converged. Then, vision is ignored, and only force is used to safely grasp and extract the book. Figure 11 shows the convergence of the visual servoing procedure.

Figure 11

Note that only two degrees of freedom are controlled during this task: the lateral motion, controlled by vision, and the forward motion which is force-controlled. Of course, the alignment task could also be done by controlling all the degrees of freedom with a visual servoing control law, as in [25]. But, in this case, more visual features would be needed, adding complexity to the problem. As the task can be done with two degrees of freedom, we have simplified visual control, by using only one robust feature which is easy to find and track in the image.

5.2. Inserting the fingertips

When contact is made, all the grasping process is guided by means of force feedback. First, the robot tries to find the space at the left of the book and to insert the longer left nail into it.

A force control law makes the robot apply a force forwards, while controlling the motion to the left. As long as the left finger lies on the book spine (Figure 12b), the robot will be applying the desired force, and it will not move forwards. When the robot detects that it cannot move forwards, it will move to the left. Thus, with the combination of this frontal and lateral motion, we make the robot follow the book surface to the left.

When the left separation is found (Figure 12c), the robot does not sense any force, and the force control law makes it move forwards until another force is detected due to contact with the right nail. During this step, the robot does not move to the left, because there are no obstacles in front.

To sum up, for inserting the left fingertip, the robot always tries to move forwards, and, when it cannot be done, it moves to the left. Following this strategy, the robot easily finds the left separation. When the robot detects that its left fingertip has advanced 2 cm with respect to the position where initial contact was made, it understands that the first nail has been inserted.

For inserting the right fingertip, the robot still tries to move the fingers forwards. If the right separation is not found, it will sense an opposite force (Figure 12d). But this time, instead of moving to the right, it opens the gripper some millimeters.

As one nail is already inserted, when opening the gripper the robot senses some lateral forces that it tries to minimize using a standard force-controller. The robot tries to minimize the force in the direction that is parallel to the book plane. Thus, this direction, along with the direction that points towards the book, are force-controlled while the gripper is opening. When the separation at the right is found, the gripper can advance freely, and thus it stops opening.

Again, when the robot detects that the gripper has advanced 4 cm with respect to the initial position where first contact was made, it assumes that both claws have successfully been inserted (Figure 12e). Simple extraction movements and regrasping complete the task (Figures 12f-12i).

Figure 12

6. Experimental results

Our system is able to navigate autonomously in a library environment, using a previously acquired map. With respect to the search of the bookcase that contains

the book, our experiments show that the algorithm normally finds the correct one. The worst case happens when the library representation is not consistent with the actual arrangement (due to some recent changes, for example). If this is the case, the robot searches in nearby bookcases, until the correct one is found. This increases the search time but, during execution, the logical library representation (mentioned in section 3.1) is updated. Hence, the next searches will be more efficient. This behavior also provides the robot with adaptation capabilities to changing environments.

The time needed for searching the correct shelf depends on the book position. It will be greater when the book is located on upper shelves. The best case is achieved when the book is on the bottom shelf of the first module. If it is not there, the robot moves to the next module, or to the previous shelf, depending on the case, thus increasing the search time. The worst case happens when the book is expected to be in the bottom shelf of the last module, but it has changed its position to the next bookcase. In this particular case, the whole bookcase is explored before deciding that the book is not on it. Once more, the library representation would be updated to avoid further problems in the future.

Regarding the linear search of the book in a shelf, times will be greater when the book is at the end. Images are captured at a resolution of 640x240. The mean execution time for segmentation and tracking is 42 milliseconds, on an AMD Athlon XP, running at 1.6 GHz. It means that the algorithms can be applied at video rate.

Times are longer when a label image must be passed to the OCR module. The total time for recognizing the text inside a label is about 130 milliseconds, which means that in one second, the system is able to fully process 7 books. Note that during the tracking, the OCR is called just with new or unrecognized labels, so that this extra time is only consumed at a few frames.

For testing the performance of the tracking algorithm, we placed the manipulator in front of a bookshelf, and moved it in a direction parallel to the books, while running the program. The arm was moved at a speed of about 1 cm/s, and the total time of each experiment was 80 seconds. Typically, 22 out of 24 books were successfully located and tracked, which gives a 91% of success. Tracking failure is due to a bad segmentation. If the separation between two adjacent book labels is not clear, due to lighting conditions or because the books are too pressed, they are

not segmented correctly and the tracker is not able to detect the boundaries and handle them.

With respect to the text recognition, a mean of 1.3 characters per label failed to be recognized. More concretely, 50% of the labels were successfully fully recognized, 30% had one or two misses, and in the other 20% more than 2 characters failed. In all the cases, the main reason for missed characters was a bad binarization and a bad OCR performance. If two adjacent characters in the binarized label are connected, the OCR software is not able to handle them. Even if the characters are clearly visible, the OCR system has difficulties to recognize them, as shown in the examples of Figure 13.

Figure 13

We hope to improve these results by switching to a commercial OCR software, and by implementing another local binarization stage, after the label has been located, and before passing it to the OCR module.

Finally, the grasping subsystem has shown to be very robust if the book did not exceed the physical limits of the system (width and weight). Friction at the fingertips is very important for any robot that has to perform manipulation tasks. In our case, the metallic fingertips have a low friction coefficient. For solving this problem, we have tested several materials at fingertips and have studied their performance. We cannot increase the friction coefficient too much, because if it is done, it can impede the insertion of the nails into the book separation. Thus, we decided to use some kind of rubber, which has more friction than the metal, but is not so high as to impede the normal development of the task. With this configuration, the UJI librarian robot can grasp most of the books. The total time for grasping a book depends on its width; an average of 1.5 seconds per 1 centimeter of book is needed. A video segment showing the system in action can be found in [26].

A final interesting issue is how much effort would be needed to adapt the system to a particular library. If the library map is available and the basic furniture structure is the same (bookcase, module, bookshelves), the corresponding user-configurable parameters can be easily modified. Then, the logical representation of the library should be stored by introducing manually the code of the first book in each bookcase; this can be done in just a few hours, depending on the size of the library. If the coding system is not the LCC, then a new function should be

defined to sort two given book codes.

7. Conclusions

Though the number of existing intelligent service applications is increasing, there are still very few that include sensor-based manipulation. We have presented the UJI Librarian Robot, a mobile manipulator that is able to autonomously locate a book in an ordinary library, and grasp it from a bookshelf, by using stereo vision and force sensing.

Instead of changing the task environment, we rather try to adapt the robot behaviour to the world and take advantage of the spatio-temporal constraints and regularities that the particular environment offers.

The robot is only provided with the book code, a library map and some knowledge about its logical structure. To accomplish its task the system integrates disparate techniques in such a way that not only each subsystem works correctly, but also the system as a whole exhibits a robust and dependable performance. These techniques include on-line image segmentation, stereo vision, visual tracking, probabilistic matching, motion estimation, probabilistic selection, multisensor-based reaching and grasping, visual servoing and hybrid control, among others.

The system has been tested, and experimental results show how it is able to robustly locate and grasp a book in a reasonable time without human intervention. On-going work aims at enhancing the robot capabilities by replacing the parallel-jaw gripper with a three-finger hand, so that book extraction is just one among many possible manipulation skills [3].

References

[1] T. Tomizawa, A. Ohya, and S. Yuta. Remote book browsing system using a mobile manipulator. In IEEE International Conference on Robotics and Automation, pages 256–261, Taipei, Taiwan, September 2003.

[2] Suthakorn, J., Lee, S., Zhou, Y., Choudhury, S., Chirikjian, G.S. “An Enhanced Robotic Library System for an Off-Site Shelving Facility”. Proceedings of the 2003 International Conference on Field and Service Robotics (FSR), Mt. Fuji-Lake Yamanaka, Japan, 2003.

[3] M. Prats, P. J. Sanz, A. P. del Pobil, E. Martínez and R. Marín. Towards multipurpose autonomous manipulation with the UJI service robot. Robotica Journal, 25(2):245-256, 2007.

[4] del Pobil, A.P., Serna, M.A., 1995, Spatial Representation and Motion Planning, Springer, Berlin.
[5] Gupta, K., del Pobil, A.P. (eds.), 1998, Practical Motion Planning in Robotics, John Wiley & Sons, New York.
[6] Bort, L., del Pobil, A.P., 2000, "Using Speech to Guide a Mobile Robot Manipulator", Proc. IEEE International Conference on Systems, Man and Cybernetics, Nashville, Tennessee, USA. pp. 2356-2361.
[7] Gupta, K., del Pobil, A.P., Choset, H. (eds.), 2000, Lecture Notes of the Workshop on Integrating Sensors with Mobility and Manipulation, IEEE Int. Conf. on Robotics and Automation.
[8] H. Choset, K.M. Lynch, S. Hutchinson, G.A. Kantor, W. Burgard, L.E. Kavraki, and S. Thrun, Principles of Robot Motion: Theory, Algorithms, and Implementations, MIT Press, Cambridge, MA, 2005.
[9] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics, MIT Press, Cambridge, MA, 2005.
[10] O. Faugeras, Q.T Luong, and T. Papadopoulo. *The Geometry of Multiple Images*. MIT Press, 2001. ISBN: 0-262-06220-8
[11] Elgammal, A., Duraiswami, R., and Davis, L. Probabilistic tracking in joint feature-spatial spaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2003.
[12] Lowe, D. Robust model-based motion tracking through the integration of search and estimation. In *International Journal on Computer Vision*, pages 113–122, 1992
[13] Schulenburg, J.. Gocr. Available on: http://www-e.unimagdeburg.de/jschulen/ocr/.
[14] M. Mason. Compliance and force control for computer-controlled manipulators. IEEE Trans on Systems, Man, and Cybernetics, 11(6):418–432, 1981.
[15] J. D. Schutter and H. V. Brussels. Compliant robot motion I. A formalism for specifying compliant motion tasks. *International Journal of Robotics Research*, 7(4):3–17, 1988.
[16] T. Kröger, B. Finkemeyer, U. Thomas, and F. Wahl. Compliant motion programming: The task frame formalism revisited. In Mechatronics & Robotics, Aachen, Germany, September 2004.
[17] J. Baeten, H. Bruyninckx, and J. D. Schutter. Integrated vision/force robotic servoing in the task frame formalism. International Journal of Robotics Research, 22(10-11):941–954, 2003.
[18] F. Marrone, F. Raimondi, and M. Strobel. Compliant interaction of a domestic service robot with a human and the environment. In *Proc. of 33rd Int. Symposium on Robotics*, Stockholm, October 2002.
[19] Y. Mezouar, M. Prats, and P. Martinet. External hybrid vision/force control. In Intl. Conference on Advanced Robotics (ICAR'07), Jeju, Korea, 2007.
[20] K. Hosoda, K. Igarashi, and M. Asada. Hybrid visual servoing / force control in unknown environment. In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1097–1103, Osaka, Japan, 1996.
[21] G. Morel, E. Malis and S. Boudet. Impedance based combination of visual and force control. In IEEE International Conference on Robotics and Automation, ICRA'98, volume 2, pages 1743-1748, Leuven, Belgique, May 1998.
[22] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in Robotics. IEEE Trans. on Robotics and Automation, 8(3):313–326, June 1992. ISSN: 1042-296X.

[23] S. Hutchinson, G.D. Hager, and P.I. Corke. A tutorial on visual servo control. IEEE Trans. on Robotics and Automation, 12(5):651–670, October 1996. ISSN: 1042-296X.
[24] Prats, M., Ramos-Garijo, R., Sanz, P. and del Pobil, A. Autonomous localization and extraction of books in a library. In *Intelligent Autonomous Systems 8*, edited by F. Groen et al., IOS Press, Amsterdam, 2004, pages 1138–1145.
[25] R.P. Horaud F. Dornaika B. Espiau. Visually guided object grasping. IEEE Transactions on Robotics and Automation, 14(4):525–532, August 1998. ISSN: 1042-296X.
[26] A.P. del Pobil, M. Prats, R. Ramos, P. Sanz, E. Cervera. The UJI Librarian Robot. In Video Proc. of International Conference on Robotics and Automation (ICRA'05), Barcelona, Spain, 2005. Available on the web: http://www.robot.uji.es/documents/videos/librarian/pobil.mov

Figure 1. The UJI Librarian Robot

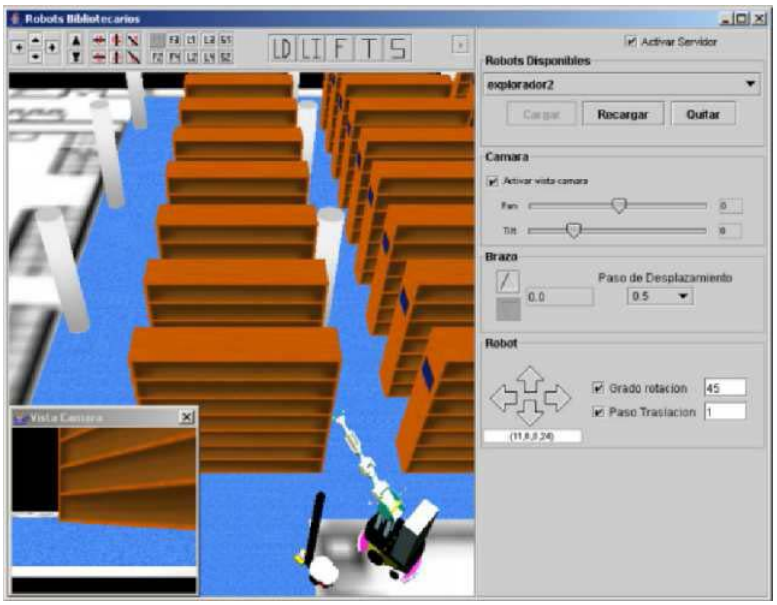


Figure 2. The operator interface

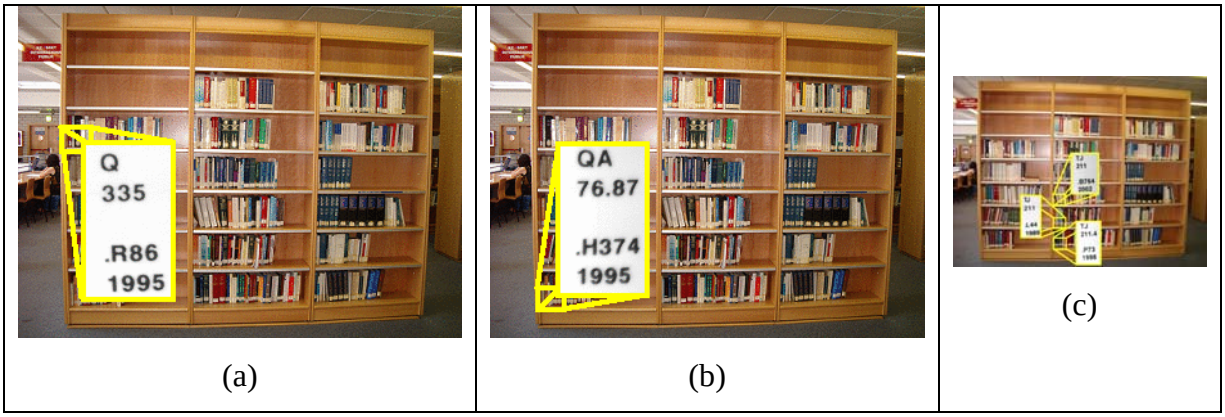


Figure 3. The book search algorithm. This particular bookcase is composed of three modules containing seven shelves each. These are all configurable parameters of the system, including their dimensions.

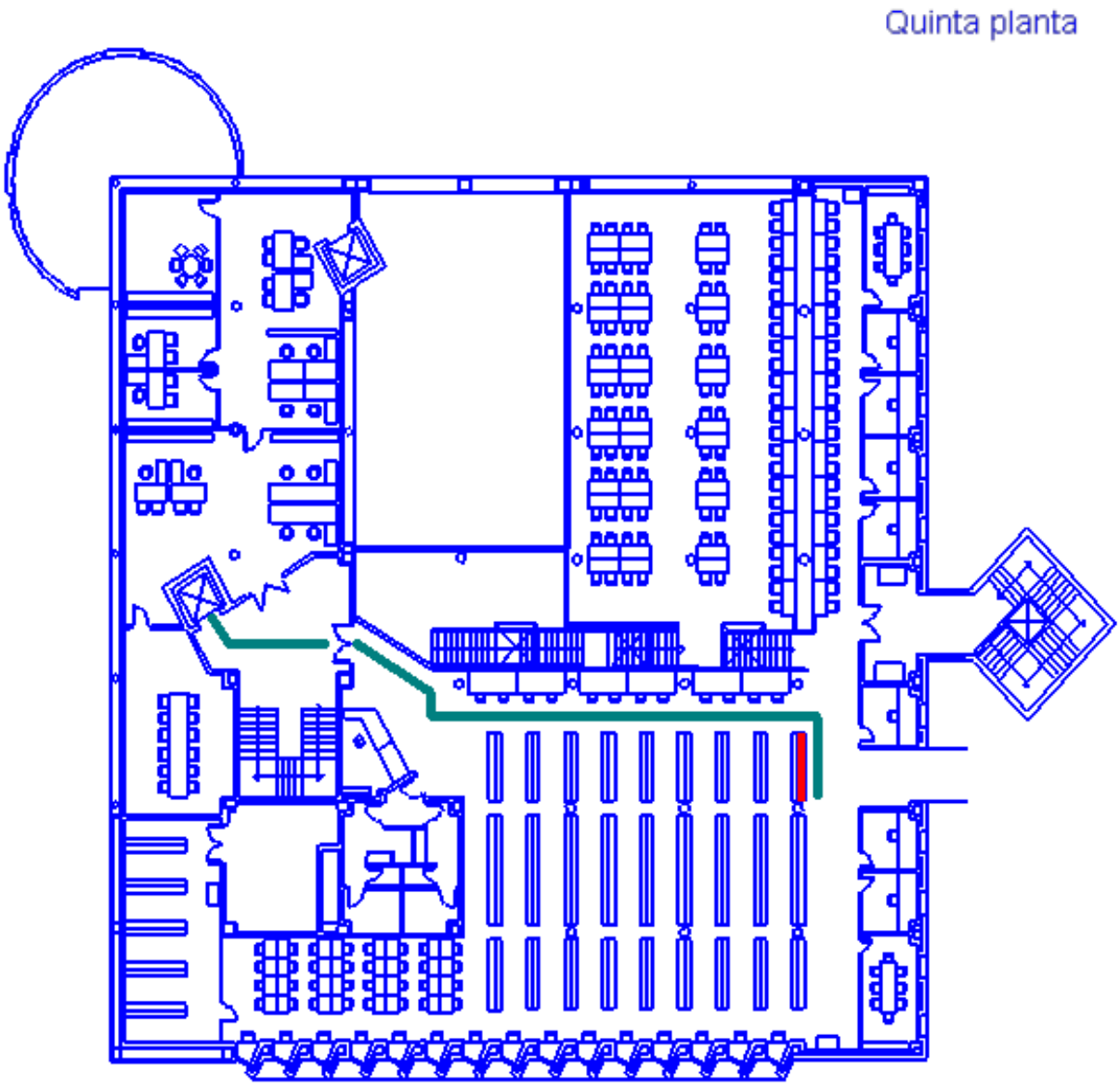


Figure 4.Library map and path planning

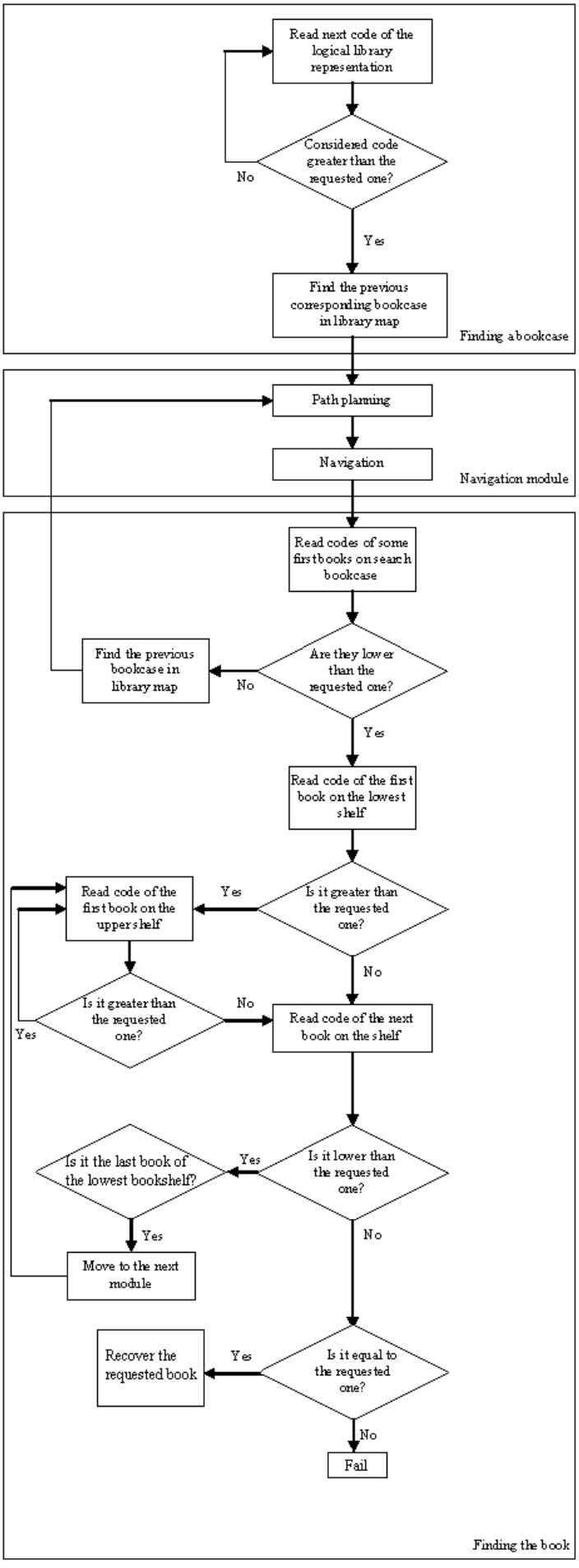

Read next code of the logical library representation
Considered code greater than the requested one?
No
Yes
Find the previous corresponding bookcase in library map
Finding a bookcase
Path planning
Navigation
Navigation module
Read codes of some first books on search bookcase
Are they lower than the requested one?
Find the previous bookcase in library map
No
Yes
Read code of the first book on the lowest shelf
Is it greater than the requested one?
Yes
Read code of the first book on the upper shelf
No
Is it greater than the requested one?
Yes
No
Read code of the next book on the shelf
Is it lower than the requested one?
Yes
Is it the last book of the lowest bookshelf?
No
Yes
Move to the next module
Is it equal to the requested one?
Yes
Recover the requested book
No
Fail
Finding the book

Figure 5. Flowchart of finding a book

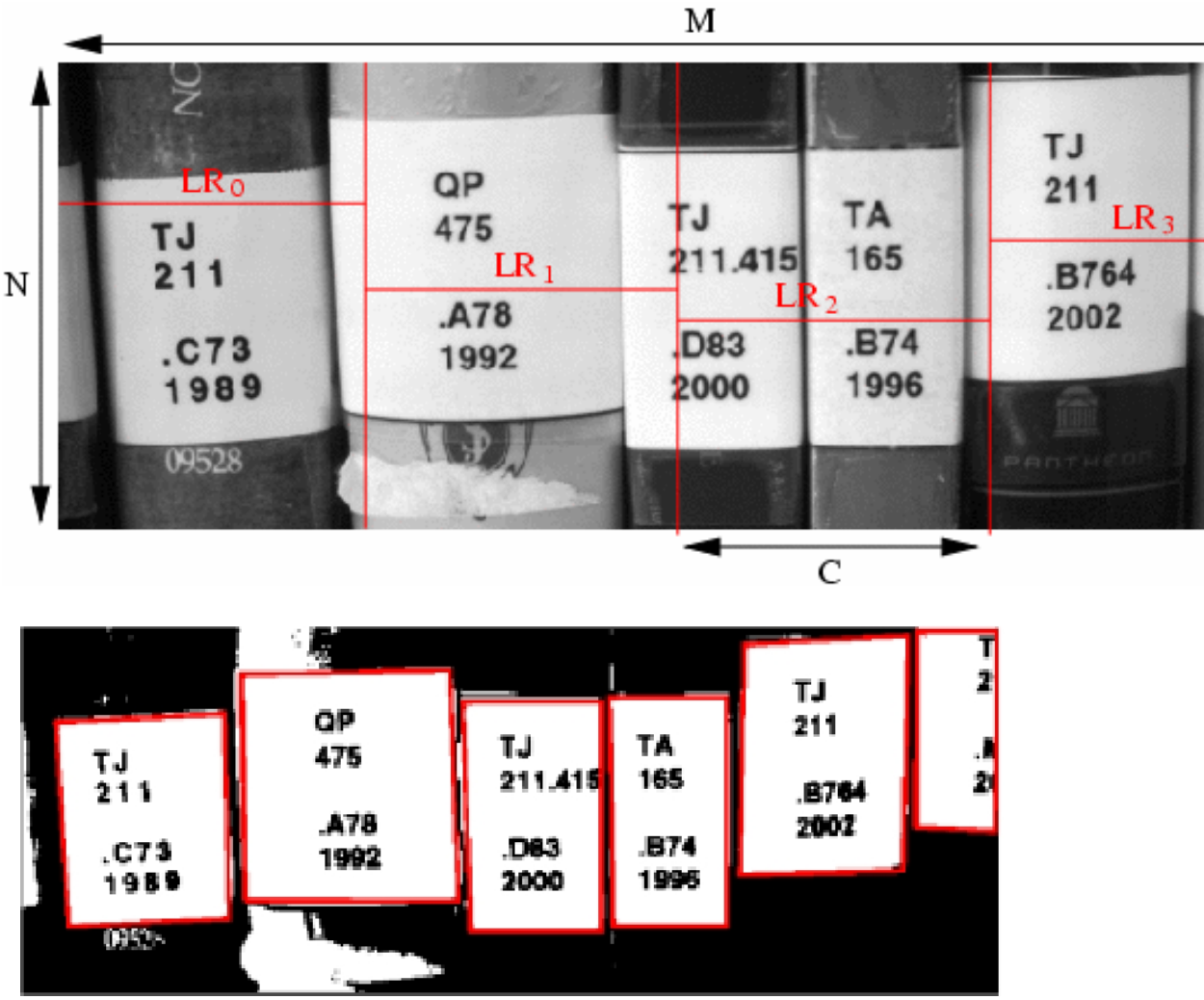


Figure 6. Label segmentation

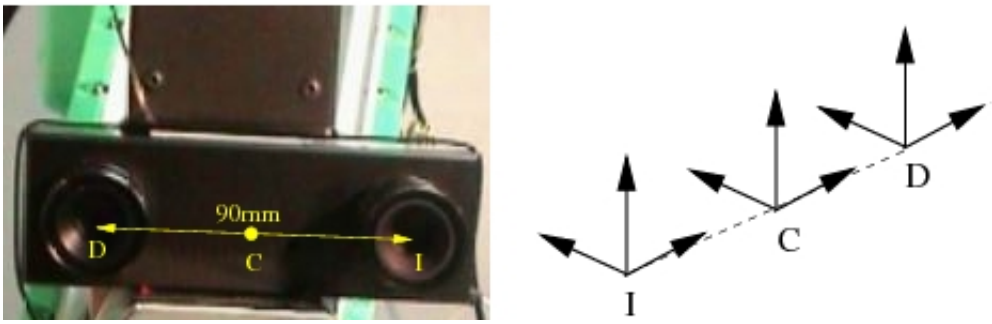


Figure 7. Camera reference systems

Figure 8. Correspondences and plane estimation

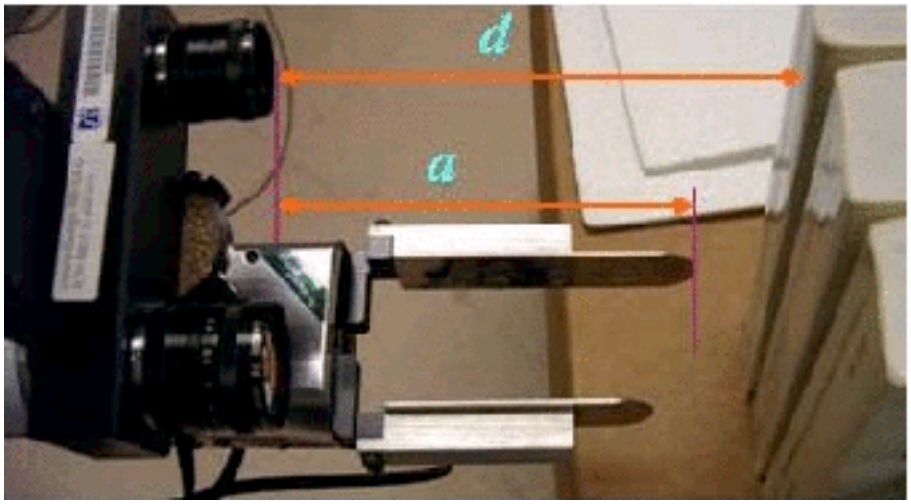


Figure 9. Vision and grasping system

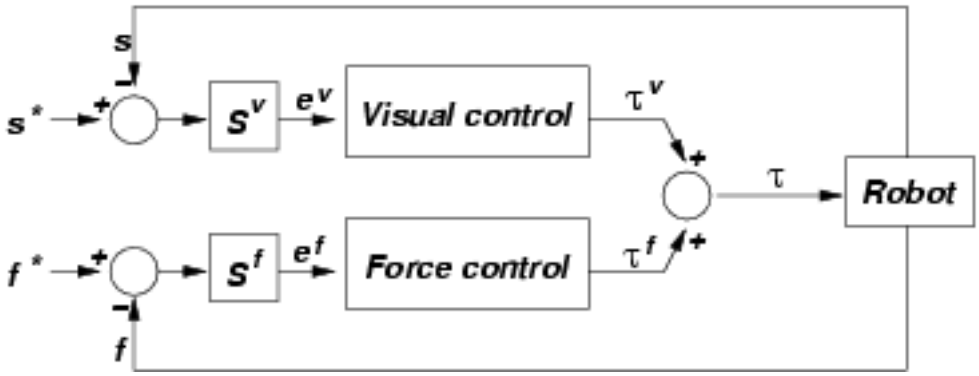


Figure 10. Hybrid vision/force control law. S is the selection matrix

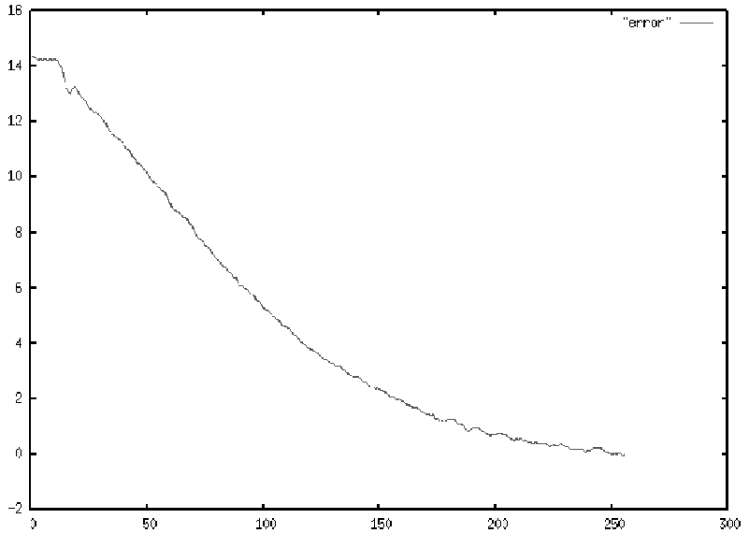
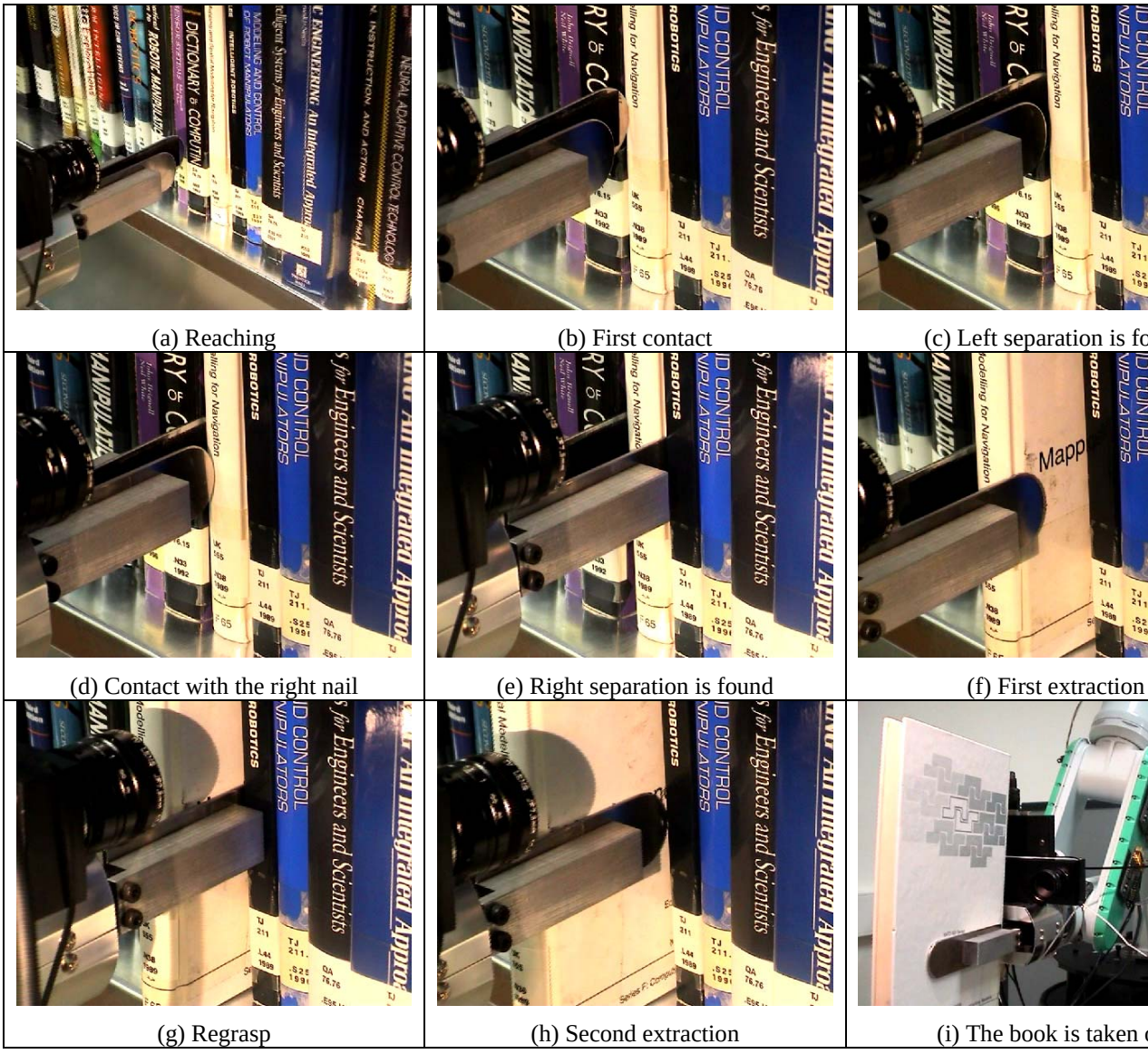


Figure 11. Error in visual servoing



(a) Reaching (b) First contact (c) Left separation is fo

(d) Contact with the right nail (e) Right separation is found (f) First extraction

(g) Regrasp (h) Second extraction (i) The book is taken

Figure 12. Steps in book grasping

QL 669.2 .I58 1987	QL 669.2 .I58 1987	QL 669.2 .I58 _987
TJ 211 .V35 1992	TJ 211 .V35 1992	T_ 2I1 .V35 1992

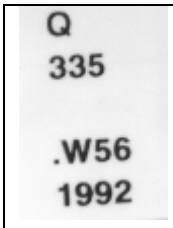
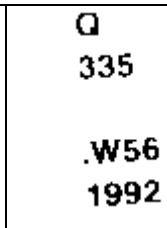
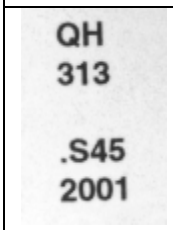
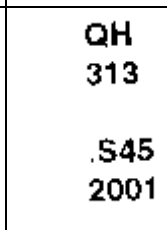
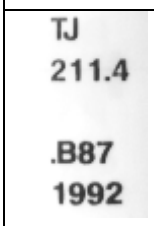
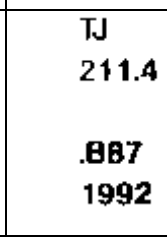
		Q 33_ ._56 1992
		QH 313 ._45 2001
		TJ 2I_.4 .BB7 1992

Figure 13. Original labels, the segmented image and the recognition results from the OCR software.