ASSEMBLY OF OBJECTS WITH NOT FULLY PREDEFINED SHAPES

M.A. Arlotti, V. Di Martino

IBM Rome Scientific Center
via Giorgione 159
Roma, Italy, 00159

Abstract

An assembly problem in a non-deterministic environment, i.e. where parts to be assembled have unknown shape, size and location, is described. The only knowledge used by the robot to perform the assembly operation is given by a connectivity rule and geometrical constraints concerning parts. Once a set of geometrical features of parts has been extracted by a vision system, applying such rule lets to determine the composition sequence. A suitable sensory apparatus allows to control the whole operation.

KEYWORDS: artificial intelligence / machine vision / robot planning / robotics / sensor integration

1. Introduction

In this paper we present an experimental work realized to investigate some robot capabilities when dealing with unstructured operational environments. Generally, different degrees of uncertainty are present in the robot operation world. In this context we could consider the following situations.

a) The shape and dimension of parts as well as the final assembly are known, while their location on the workplane is unknown. In this case the robot vision system must recognize parts and determine their location and orientation. Then the robot has to plan the composition sequence to reach the final assembly. This implies to define a grasp approach trajectory for the arm, a grasp position for the end-effector, a "collision-free" trajectory and a suitable positioning of the moved part into the assembly to be built. Use of endpoint sensing should be made by the robot in an interactive fashion in order to recover unpredictable error situations.

b) The shape and dimension of parts as well as their locations are not known while the final assembly is. In such a case the vision system must locate the various parts while checking, starting from a general knowledge of the problem, if they are admittable parts to be assembled or not, however without identifying them, since they are not completely known "a priori". In addition to the items examined in the previous case, the planning effort implies to determine an assembly sequence of the given parts that matches the goal. Some constraints are to be considered in order to make possible a solution strategy.

c) As an extension of the previous case, not only the shape, dimension and location of parts are unknown but also the assembly goal, meaning that the only "a priori" knowledge consists in a set of possible goals. Besides, some assembly constraints and rules should be considered. Once

parts have been located and a number of features of them have been extracted by means of vision, the planning system, using the given constraints and rules, must try to match the various possible goals with the given parts.
The described situations correspond to different philosophies for the use of a robot in manufacturing environments. The former is the most usual in industry: parts are known together with the way they are to be assembled. A large amount of *a priori* knowledge mitigates the problem complexity, both for visual recognition and for planning. Conversely, the latter correspond to a case where a number of parts are present and the robot ignores what assembly they belong to. This could be useful in flexible environments (FMS-FAS) where a mix of products can be handled at one time. Obviously, the vision apparatus should give more detailed and accurate information and the planner has to solve a more complicate problem. At the moment there are no industrial applications realized to operate in such a way, primarily because they are not cost effective.
Our experiment has been carried out considering a very simple assembly case. The purpose has been to validate some robot reasoning capabilities in a practical problem. The problem configuration has been such to neglet other essential planning issues, in particular collision avoidance and grasp planning, considered in previous experiments [1][2] .

2. Experiment description

The task the robot must perform is to compose a plane figure starting from some pieces placed on its operation plane. Such pieces are made by a white thin millboard and are placed on a black background. This choice semplifies the vision effort while complicating the planning complexity, as will be explained later. The pieces have been obtained by cutting a millboard polygonal figure, choosen among a set of some similar ones prestored into the robot memory, by means of random straight cuts. Thus the pieces are intrinsically unknown "a priori" to the robot system. Then, such pieces are randomly located onto the operation plane by the oper-

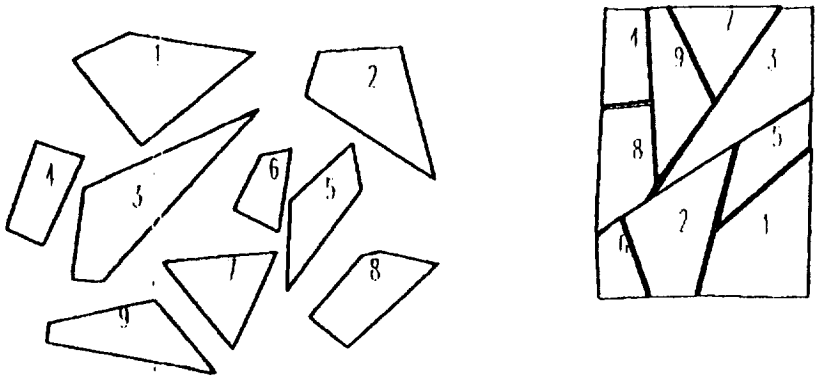


Figure 1. Puzzle assembly example

ator. Obviously, in order for the robot system to find a solution requires that all pieces are consistent with one of stored figures, i.e. all of them should have been generated cutting one such figure. In other words, the robot task consists in manipulating pieces ignoring their number, shape, dimensions, location and what assembly they belong to. It could be considered a *generalized puzzle problem* (fig. 1).

In addition to the previous uncertainties, it should be considered that some pieces can be placed onto the robot plane *overturned* with respect to the upper face of the figure. It is important to note that the millboard has both faces white, so that the *up* and *down* faces of each piece are indistinguishable by means of visual information, tipically by colour. Then, it is only by means of reasoning that the robot should be able to detect a similar situation identifying the initial figure and assembling all pieces in the correct way.

As a preliminary step, the robot must learn the figures it has to reconstruct and store them into the robot memory. Each figure, after beeing placed onto the robot plane, is acquired by the robot camera, then coded into numeric information to be stored in a suitable database. Once some figures have been stored in this way, the robot is able to compose any of them if some pieces are placed, by the right side or overturned, onto the operating plane. As explained above, the only constraint in order for the system to reach a solution, is that the various pieces must be consistent with one figure. In case of inconsistency (the pieces do not belong to anyone of the stored figures or they are less than needed), the system tries to compose a default stored rectangle. If this planning attempt is also unsuccessfull the system fails notifying the event to the operator.

Two further constraints must be considered, regarding the generation and location of the pieces involved in this experience. In particular: a) The pieces must be generated by straight side-to-side cuts: this implies that all pieces are convex polygons and is required by the geometric reasoning procedure. b) The pieces cannot be placed *overlapped*. This should be immediately clear observing that no piece is known "a priori" by the system and then only a complete visual information allows a complete knowledge of its required characteristics. In short, it should be taken into account that the experiment was designed to validate an AI approach to a concrete assembly problem, leaving unsolved some practical issues.

3. Adopted approach

The solution of the described problem is based on a geometrical reasoning approach, since this perfectly matches the problem characteristics. In order to implement the reasoning process it is necessary to traduce the various pieces to assemble into a set of geometric elements. This is accomplished during the vision process, when the image of the various assembly elements is traduced into a sequence of *vectors*, by a process called *vectorization*. Each vector is the representation of an edge side of a polygonal piece. The geometric reasoning operates on such vector sequences applying some geometrical connectivity rule in order to find the assembly sequence (solution). Once this has been found, it is necessary to traduce it into phisical displacement/rotation pairs, in order to perform a correct manipulation at assembly time. So, three classical activity steps can be identified: vision, planning and manipulation. Let us examine separately each one of them.

3.1 Vision

The vision task consists essentially in a low-level phase, by which the polygonal pieces boundaries must be located and traduced into vector sequences. No recognition is made since pieces are unknown. It operates according to the following steps:

- image acquisition and binarization using a suitable threshold
- edge detection by means of a raster-to-vector conversion
- vector postprocessing, to eliminate vectorization artifacts

The raster-to-vector algorithm [3] consists essentially of three phases. (i) A *pre-processing* step, consisting in filling gaps and removing noise from the raw image. (ii) A *boundary tracing* step, which consists in determining boundary points between binary regions. (iii) A *line following* phase, where segment-like regions are transformed into couples of points coordinates (extremes). The raster-to-vector conversion, because of the discrete nature of the image, can be affected by some errors, such as unexisting short sides or adjacent sides with very similar orientation (see fig. 2). Very often such cases are conversion artifacts corresponding to the following phisical situations. (a) A boundary corner is not "seen" as a real tip but as a confused edge region, so that it is converted to a short vector, instead of the cross point between two adjacent vectors. (b) A side, because of its bending, due, for instance, to lens distorsion, is broken into two sides with very similar orientation. In order to filter such artifacts a postprocessing phase has become necessary.

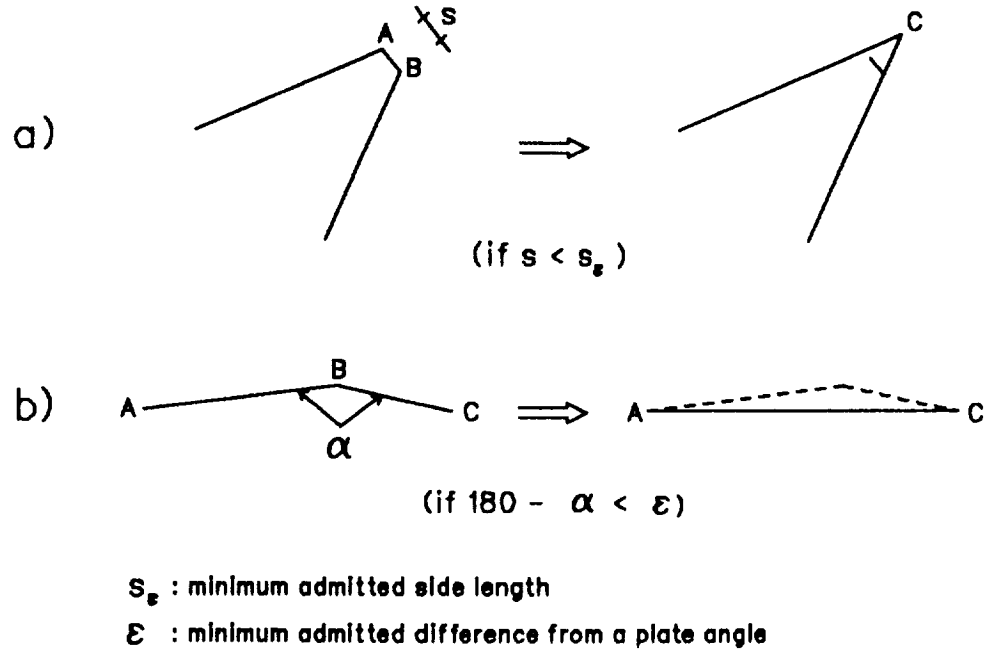


Figure 2. Contour extraction: vector postprocessing

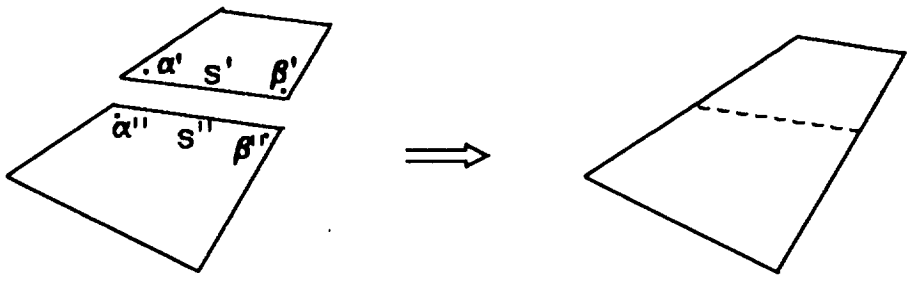
Essentially, this operates as follows (see fig. 2):

- vectors too short are eliminated lengthening the two adjacent ones in the sequence until they intersect;
- angles between vectors very near to 180 degrees are eliminated rectifying its sides to obtain a unique vector.

In addition to the previous artifacts, very small white regions can be detected and converted; usually they correspond to spots due to manipulator oil leakage. They produce closed vector sequences with a very small enclosed areas: for each one the postprocessor tests the area value and, in case this is less than a given threshold, erases the whole vector sequence from the world state.
Hence, this phase solves the most vision problems. Anyway, the final vision data (initial world state) are affected by some amount of precision error, due to lens distortion, camera calibration, image resolution, conversion quantization.

3.2 Process planning

As stated before, the "world"representation built by the vision is not completely accurate. This fact should be taken into account by the assembly solution method. For such a reason, an error insensitive connectivity rule is adopted. The global solution strategy is based on the recursive application of such a rule and operates on reduced search spaces. [4] This means that intermediate world states are created during the solution search.
In the following by "polygon" will be denoted a generic piece. The adopted rule allows to determine when two polygons are adjacent along a side in the recomposed figure. In particular, it states what follows: "two polygons are produced by one cut if they have a side of equal length and the angles at the extremes of this side supplementary two by two" (fig. 3). In this way the solution method consists in comparing three couples of values for each couple of sides (one



$$\alpha' + \alpha'' = 180$$

$$\beta' + \beta'' = 180$$

$$s' = s''$$

Figure 3. Connectivity rule and world state update

couple of segments and two couples of angles). Each comparison is made with a prefixed tolerance to take into account errors introduced by the vision system. This makes the process quite error insensitive. The rule is applied to all couples of sides by an exhaustive search among all polygons of the actual world state. When a couple of sides, belonging to different polygons, satisfies the connectivity rule, the planner "adds" the polygons along the common side, building a new abstract polygon and deleting the previous two from the world (fig. 3). This corresponds to update the world state at each recursion. When a unique polygon remains in the world (final world state of the planning process), this is compared with each of the initially stored figures, which are the goal of the planning process. When such a polygon matches, with some prefixed tolerances, one figure in the database, the process is successfully ended and the figure is identified.

Anyway, the process could "fail" at any step. If this happens before a unique polygon has been assembled, it implies that the rule fails for all the couples of polygons actually in the world. This means that an inconsistent set of pieces has been submitted to the robot. Conversely, a fail could also occur when a unique polygon remains in the world state. Generally, this sholud be ascribed to wrong adjacencies, i.e. to couples of sides satisfying the connectivity rule but not arising from physical cuts. Wrong adjacencies are then marked not to repeat, during following searches, wrong branches of the research tree (this mechanism is commonly called *backtracking*). Finally, when all the research tree has been visited without any success, the system identifies an inconsistent set of pieces.

The described problem solver corresponds to an initial implementation. Next, in order to detect overturned polygons on the scene, an enhanced problem solver has been implemented, where the connectivity rule is applied to all polygons in right and overturned configuration (the sides-angles sequences are inverted), tracking for each successful operation the initial condition (right or overturned) of each elementary polygon. This method increases the number of wrong adjacencies and so, requires smaller tolerances. As expected, it results more time consuming than the previous one.

Hence, the result of the entire problem solving process can be:

1. Solution found with right pieces: list of polygon adjacencies
2. Solution found with some pieces overturned: list of polygons to be overturned
3. Solution not found: pieces inconsistent with all the initial figures

As mentioned before, the problem solving process is implemented by two different software modules. The first one is capable to find a solution only when polygons are in right position and can be immediately assembled; it is characterized by a quite fast execution time. The second one, started only when the first fails, can recognize a more complex situation, discriminating between the case of overturned polygons and that of pieces inconsistent with the initial figure database. The latter is of course, much more slower than the former. This software architecture, based on two distinct problem solvers with different capabilities, has been mantained in order to optimize the performance of the whole planning process.

Both modules are written in Prolog language because of the "built-in" backtracking mechanism of such a language.

The planning output, i.e. the list of polygon adjacencies, in order to be used during the physical manipulations, is traduced into a sequence of couples translation + rotation, needed to the manipulator to displace the pieces. This is made by a suitable sequential program, starting from the knowledge of initial location (from vision) and final position (outside the camera field, then predefined) for the whole assembly. An analogous conversion is made for the pieces to be overturned, taking into account the effects of the upsetting operation at manipulation time.

3.3 Manipulation

Starting from the sequence of the displacements and rotations, the actuation module controls the physical handling of the pieces. The puzzle pieces are millboard plates, randomly distributed on the work plane inside to the visual field of the tv camera, while the reassembled figure is constructed by the manipulator on an area outside such a field. The whole pick-and-place of a single piece is made by a particular lifting actuator, a suction cup, which is grasped and held by the manipulator gripper (fig. 4). Such particular actuator is driven in *on-off* mode by the robot controller in order to grasp/release the piece itself. Its operation principle is based on a Venturi tube which generates, when air flows through it, the necessary vacuum for it to operate.
Two critical phases during the described operation are identified. First the picking/release of a piece, the complementary steps where the actuator approaches the workplane surface. In both cases it is necessary to control the motion using the tip force sensors of the gripper to detect the impact with the plane. These are continuously monitored: when the sensed impact reaction force overcomes a given threshold, the motion is stopped and the actuator is switched, on/off, depending on the operation to be performed (picking or releasing). The second critical operation is the upsetting of an overturned piece. This is obtained by means of an experimental fixture realized *ad hoc*, consisting in a kind of vice with two couples of elastic jaws, devoted to

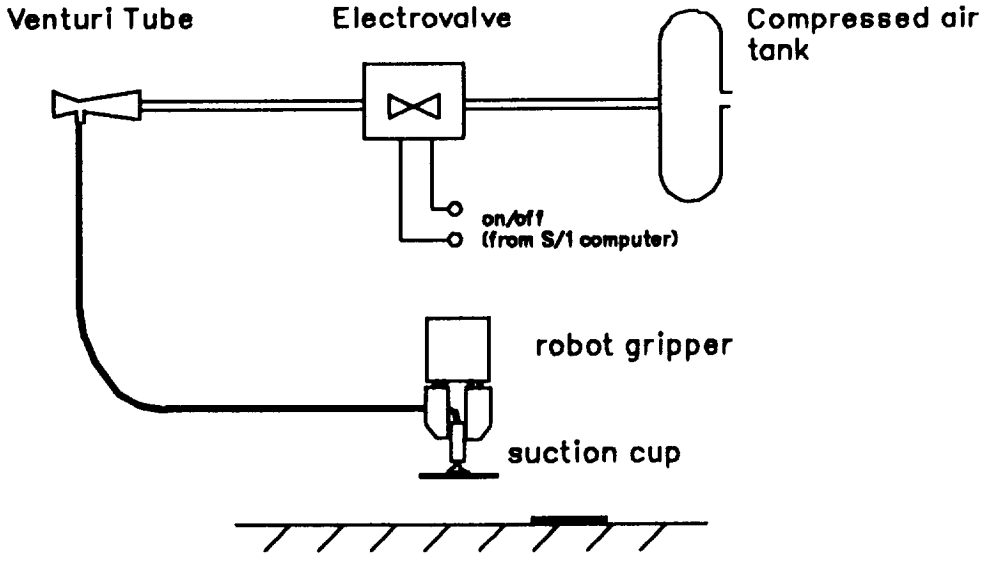


Figure 4. Pickup arrangement

hold the piece to upset while the actuator approaches it by the opposite side. Such structure of the fixture allows the actuator to release the piece, invert its orientation with respect the piece and get it back. The last step is critical. In fact, in order to have a reliable hold of the piece, it is necessary to approach it with a sufficient pressure without deforming it. To obtain this, the actuator must "search" the piece using the tip force sensors.
It should be remarked that sensors are also used to control the actuator grasping. Normally the actuator is fixed on the workplane, in a known position. The gripper approaches it, verifies its real presence by the presence sensor and grasps it controlling the tightening force by the pinch force sensors.

4. Hardware configuration: the robot workstation

The described experiment has been carried out on general purpose robot workstation set up at IBM Rome Scientific Center. This is based on a IBM 7565 robot, which is controlled by a special version of the IBM S/1 minicomputer, integrated with some other machines and computing facilities in order to achieve an adequate power so as flexibility, to develop similar experiments. [2]. Figure 5 shows the overall workstation architecture. The whole station supervision is performed by a personal computer AT. Furthermore, it implements the user

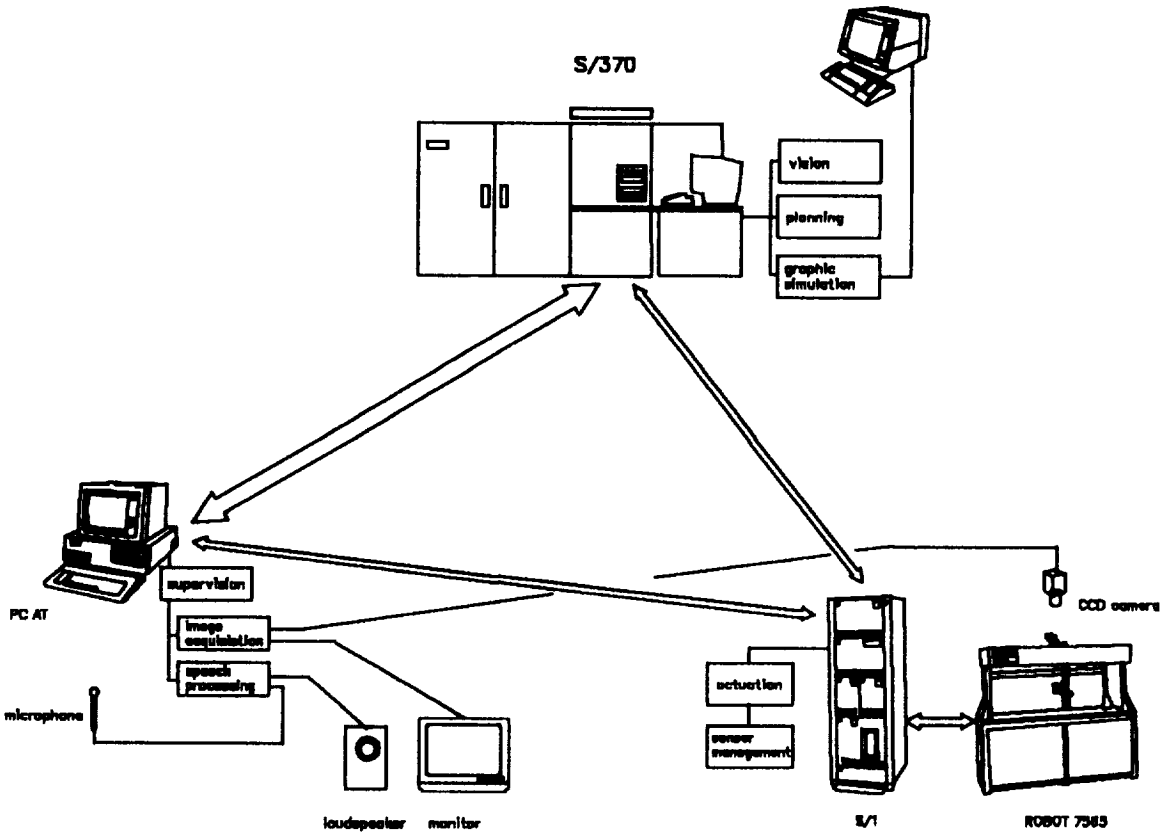


Figure 5. Workstation architecture

interface acting as a system console. Such interface makes use of a speech recognizer and a speech synthesizer. In addition, the PC is devoted to image acquisition and preprocessing for robot vision tasks. The station includes also a S/370 mainframe which is used to perform hard computations such as machine vision, planning tasks and graphic simulations. The three mentioned computer systems are connected together in a network with triangular topology and bidirectional links. In particular, the PC and the mainframe are connected through a S/370 channel attachment to have a fast transfer of large image data sets. The other network links are serial lines, being devoted to more concise data set transfers.

4.1 The manipulator

The IBM 7565 [5] is a cartesian hydraulically powered manipulator, consisting of 6 d.o.f. arm supported by a parallelepiped box frame. Its joints, three prismatic (arm joints, X,Y,Z) and three revolute (wrist joints, *roll, pitch and yaw*), are controlled by analog position servos driven by the robot controller. The gripper is mechanically configured so that the finger surfaces move toward each other remaining parallel. A set of endpoint sensors are mounted in connection with them: three couples of force sensors and a presence sensor. The former are *strain gauges* connected, for each finger, along the three spatial directions. The latter consists in a led-phototransistor pair (*led-beam*) which, once broken by an opaque object located between fingers, lets the manipulator to detect its presence.
The described manipulator is programmed by a special purpose language called A.M.L. ("A Manufacturing Language") [6]. This provides an interactive environment to perform robot motion control, sensor management, data processing and data communication. In the AML environment two different modes are available to process sensors signals. The first one is under program control: sensors are polled and tested by the application program. The second one is an asynchronous, interrupt-like mode; this means that it is possible for the system to detect sensory events (force threshold overcoming, led-beam interrupt ...) in an asynchronous way, interrupt the running AML program at any instant and run a proper user-written AML service routine.

4.2 Image acquisition subsystem

The image acquisition process involves many different hardware and software components. The image is acquired using a CCD camera fixed over the robotic scene and looking downward with the optical axis perpendicular to the robot plane. The camera is attached to the PC via a frame grabber with a resolution of 512 x 512 pels. The acquired image is monochromatic with 256 gray levels: such features have appeared to be adequate in the most 2-D vision experiments carried out until now. In the actual experiment the chromatic resolution is not a critical point, beeing the image thresholded and reduced to a bitmap.
A critical feature of the acquisition is *camera calibration* i.e. the knowledge of the correct correspondence between the world (robotic plane) and the camera coordinate system. First, due to CCD sensor geometry (rectangular) and to the grabbing process (producing a square image), the real scene and its image are not isomorphic. In other words, the spatial passes corresponding to

pixels in the horizontal and vertical directions are not the same. To overcome this problem, the image is stretched in the horizontal direction, by an experimental *stretching factor.*
Besides, the two mentioned coordinate systems have not the same origin, orientation and scale. Thus, to transform a coordinate pair to another it is necessary to determine the proper transformation parameters: this is the goal of the calibration process. Normally, this is made by a linear process, by sensing two different reference points (*calibration posts*) in robot coordinate (mm) and in image coordinate (pels). Such values are used in a linear equation system, whose solution are the reference system change parameters (x_o, y_o (mm) of image origin and k_x and k_y, ratios between pels and mm along x and y). Such process does not take into account the non-linear behavior of lens near edges. This gives an acceptable accuracy in applications not requiring a high precision, while in other applications, like the described one, this is not acceptable.
More accurate results have been obtained applying the same procedure to various couples of posts, placed simultaneously in different points of the scene. For all the couples the required parameters are computed; in this way for each parameter a sample of values is obtained. For each sample mean and mean square error are computed; the final value of each parameter is determined discarding those values outside m.s.e. and computing the mean of remaining. This procedure gives a more "robust" calibration mitigating the effects of lens distorsion. A very accurate calibration procedure is described in [7].

Bibliography.

[1] R. Golini e M. Arlotti **An Intelligent Robot Workstation,** Artificial Intelligence - Implications for CIM, IFS / Springer-Verlag, 1988.

[2] M.A. Arlotti, A.De Castro,V. Di Martino,C.Raspollini,M. Vascotto, **Applicazioni di robotica e visione: esperimenti di manipolazione,** *Robots between Science & Technology SIRI fourth National Conference Proceedings, 211-226.* Milan, March, 21-23, 1988.

[3] IBM Japan Science Institute **Graphical Image Formatting and Translating System User's Guide** Tokio, 1985.

[4] V. Di Martino **Algoritmi di pianificazione per robot** *Atti delle giornate AIRO* Pisa, october 1988.

[5] IBM 7565 Manufacturing System, **Maintenance Information, Circuit Diagrams and Schematics, and Installation,** 1983.

[6] IBM 7565 Manufacturing System, **A Manufacturing Language Reference,** 1983.

[7] R.Y. Tsai, **An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision,** IEEE Conference on Computer Vision and Pattern Recognition, Miami, 1986.

ROBOT KINEMATICS, DYNAMICS AND CONTROL