A Visual Servo

Nielsen, Lars

1987

Document Version:
Publisher's PDF, also known as Version of record

CODEN: LUTFD2/(TFRT-7339/1-010/(1987)

A Visual Servo

Lars Nielsen

Department of Automatic Control
Lund Institute of Technology
February 1987

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
Report

Date of issue
February 1987

Document Number
CODEN: LUTFD2/(TFRT-7339)/1-010/(1987)

Author(s)
Lars Nielsen

Supervisor

Sponsoring organisation
The Swedish Board of Technical Development

Title and subtitle
A Visual Servo

Abstract

Control based on image information offers great possibilities for advanced automation. Here, a flexible system for study of visual servoing has been developed by extending a stnadard computer with commercially available instrumentation. A robot moving on the floor is controlled by image feedback. The conclusion is that an industrially useful behavior of automated guided vehicles (AGV) in flexible warehouses could be obtained using available technology and the principles presented here. A flexible system increases the requirements on good interaction with an operator. A man-robot interface has been designed, based on operator manipulating interactive color graphics overlaid on images of the working scene of the robot. The interaction is done specifying obstacles and stations rather than explicit path coordinates. Visual recognition in a three-dimensional scene has to cope with the problem that objects have infinitely many poses, which may not all be stored. A key contribution in this work is the design of marking symbols, which are described by invariants under parallel projection. The invariants are based on integrated measures and are thus insensitive to individual pixel errors. The constructed symbols can be used for robot marking or as signposts. The marking problem formulation is new and so are the results and the analysis.

Key words
Image processing, Vision, Feedback control, Robotics, Event handling, Motion programming, Image invariants.

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
English

Number of pages
10

Recipient's notes

Security classification

Paper presented at the 1st IFAC Workshop on Vision Control,
June 10 - 12, Espoo, Finland.

A VISUAL SERVO

Lars Nielsen

Department of Automatic Control, Lund Institute of Technology,
Box 118, S-221 00 Lund, Sweden

Abstract. Control based on image information offers great possibilities for advanced automation. Here, a flexible system for study of visual servoing has been developed by extending a standard computer with commercially available instrumentation. A robot moving on the floor is controlled by image feedback. The conclusion is that an industrially useful behavior of automated guided vehicles (AGV) in flexible warehouses could be obtained using available technology and the principles presented here. A flexible system increases the requirements on good interaction with an operator. A man-robot interface has been designed, based on an operator manipulating interactive color graphics overlaid on images of the working scene of the robot. The interaction is done specifying obstacles and stations rather than explicit path coordinates. Visual recognition in a three-dimensional scene has to cope with the problem that objects have infinitely many poses, which may not all be stored. A key contribution in this work is the design of marking symbols, which are described by invariants under parallel projection. The invariants are based on integrated measures and are thus insensitive to individual pixel errors. The constructed symbols can be used for robot marking or as signposts. The marking problem formulation is new and so are the results and the analysis.

Keywords. Image processing, Vision, Feedback control, Robotics, Event handling, Motion programming, Image invariants.

1. INTRODUCTION

Recent developments in imaging technology are substantial and exciting. Research on the applications of this technology to the field of automatic control is of significant interest. One particularly interesting application is automated guided vehicles (AGV) used in warehouses as part of a flexible manufacturing system. Currently these vehicles are controlled by magnetic trails in the floor, and the paths are essentially fixed due to the cost of rearranging the magnetic trails. A much more flexible system can be developed using a visual servo with a camera on a wall or ceiling supervising the AGV. Of course hybrid systems with visual servoing in specific areas and magnetic trails for traveling in between can be used in warehouses where flexibility is not needed everywhere.

The scene is three-dimensional and the shape and the size of objects thus vary when viewed in different orientations and at different distances from the camera. Current industrial vision systems are essentially limited to the interpretation of images extracted from a primarily two-dimensional scene. The approach is based on edge detection and contour description. Special purpose hardware implementing this approach is commercially available and typically used in industrial applications involving conveyor belts (Petersson, 1983). The extension of this technique to the interpretation of images extracted from a three-dimensional scene is a major research direction and promises to increase the applicability of the existing state of the art hardware.

The theory of visual servoing has not developed far enough so that the closed loop performance can be judged only by analysis of theoretical models or simulations. A prototype system therefore needs to be built, and significant experiments with the system must be performed. Here the philosophy has been to use simple robots and develop a flexible laboratory at a reasonable cost. The laboratory set-up is presented in Section 2. In Sections 3 and 4 the abstract structure of the AGV scenario is formalized to clarify the aspects of the system common with general robot applications. Section 5 presents the man-robot interface. Our approach to visual recognition in three-dimensional scenes is presented in Section 6. The key idea is to use marking symbols, which have properties invariant under projection. Section 7 gives some experimental data, and conclusions are given in Section 8. Throughout the presentation a number of details have been left out. A complete presentation can be found in (Nielsen, 1985).

2. THE LABORATORY SET-UP

The laboratory represents the first effort in vision control at Lund University. The project started 1982, and has included development of both hardware and software. The image system is built around a Matrox raster image memory interfaced to a VAX-11/780 via a Unibus-Multibus-interface. Robots and interactive devices are interfaced to the Vax via general AD and DA converters. Two joysticks and a mouse with three on/off buttons are used for interaction. A schematic overview of the system is given in Fig. 2.1.

2.1 Hardware

The video camera used is a black and white Intensa GPC-25 camera. The camera can be equipped with any tube of the 1" vidicon type and any lens with a C-mount (1"x32

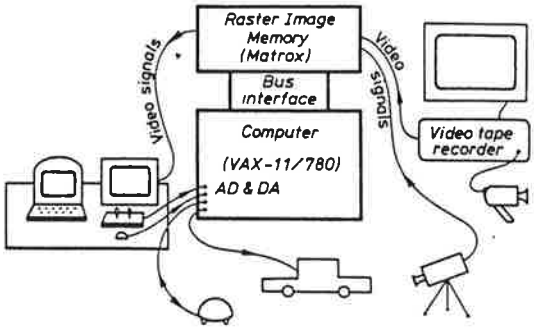

Fig. 2.1 A schematic overview of the system that consists of different pieces of standard equipment. The interfaces are simple and general.

threads/inch). In the laboratory set up the camera is equipped with a 1" newvicon camera tube. A 25 mm lens with automatic aperture control is used. The resolution of the camera is approximately 525 lines and 800 dots per line, the light sensitivity is 1-2 lux, and the automatic light compensation is 1:30000. A video camera like ours equipped with a high sensitivity vidicon combined with automatic aperture control solves the problem of getting good visual quality of the image data, when the illumination changes. The camera may without any adjustments be operated in illuminations varying from outdoor summer season daylight to indoor standard lamps. A Barco CD 33 HR RGB monitor is used to present the video output. There is also a home video set with a video tape recorder, a color video camera, and a TV. The color video camera of the video set has lower quality than the GPC camera. The video set provides, however, an easy way to handle image information from other places via the exchange of video tapes.

A raster image memory with simple interfaces (both for the video and the computer) has been built from commercially available plug-in boards manufactured by Matrox. The image memory uses standard video-signals to external video equipment such as video-cameras, video-recorders, or monitors. The connection to the computer is via a general bus. The bus communication bandwidth is approximately 1.6 Mbyte/s. Two RGB-Graph/64-4 boards provides the system with a raster memory of 512 x 512 pixels with 8 bits/pixel, where each board stores 4 bits/pixel. The VAF-512 board is a video input/output processor board. The video hardware works at the video rate of 25 frames per second. The boards are controlled via programmable I/O registers. The controls include the timing of the image system (it can either follow an on-board sync generator or follow the input video signal via an on board phase-locked loop circuit), the frame grab commands (selection of input channel, of repeated image sampling or frozen image, and of gain and offset values in the video quantization), the read and write of the content of the memory (random access of individual pixels), and the video output control interpreting the content of the raster memory via a programmable color look-up table. The VAF-512 board provides the master sync pulses of the image memory, the camera and the RGB monitor.

Any process may be interfaced via the AD and DA converters. In our experiments two vehicles moving on the floor are used. They are seen in Fig. 2.2. One vehicle is a commercially available toy Turtle marketed by Terrapin, Inc. The Turtle has a dome which is 30 cm in diameter. It has two separately driven wheels individually controlled using on/off signals transferred via a cord. For each wheel the drive train has an electrical motor and a gear-box. The motor makes 300 revolutions for one revolution of the wheel providing a maximum speed of roughly 15 cm/s. The Turtle has touch-sensors. If the dome of the Turtle is tilted due to the contact with an object then the status of four switches indicates one of eight directions. The control system of the Turtle may thus combine image and touch information.

The other vehicle is the Ilon car, which has four Ilon wheels. It allows complete control of the motion in the plane via control of the rotation of the four wheel motors. The Ilon car may thus be moved straight forward, straight sideways or be rotated on the spot. The on-board electronics of the Ilon car is controlled using three analog voltages in the range −10 - 10 V, transferred via a cord. They are set points for the speed of the forward-backward motion, the sideways motion, and the rotation motion. The dimensions of the car are: length 60 cm, width 25 cm, and height 20 cm.

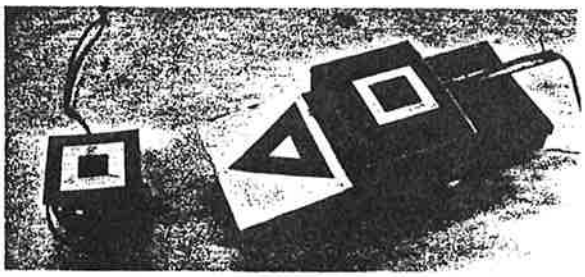

Fig. 2.2 Photograph of the two vehicles. The Turtle is seen to the left and the Ilon car is seen to the right.

2.2 Software

We foresaw large programs when the project was started. Therefore, the idea of object oriented programming was adopted. This means that the code can be modularized in packages so that types, variables, procedures which are logically connected, can also be kept together textually. Programs are built by having packages performing basic functions and then by developing packages of increasing levels of abstraction based on the basic functions.

The lowest level software is a package containing routines for communication between the raster memory and a program on the Vax. The registers of the raster memory are mapped into addresses of the Vax virtual memory, and the functions are performed via the information passed to these registers. The routines are implemented in assembly language. We have chosen to use the programming language standard Pascal for the further abstraction levels of the code. A file handling preprocessor has been developed to modularize code while keeping to standards. Packages are implemented as text files with a special structure, containing Pascal statements and preprocessor keywords. These keywords are commands to the preprocessor which divides

the file into sections corresponding to the sections of a Pascal program. The sections are then combined into a program. A style inspired by Ada has been adopted, where packages are structured in terms of a specification and a body.

There are a number of basic support packages implemented in Pascal. They contain sufficient operations to build user programs without knowing the explicit organization of the hardware. The operations include video control commands, image definitions and in/out operations, color look-up table handling, AD and DA conversion, and file handling to save and restore images. The virtual memory of the Vax automatically solves the problem of addressing large amounts of data in software. It is thus neither necessary to build special hardware nor to make complicated file handling to deal with images. A complete example in (Nielsen, 1985) describes how to make a simple but complete user program based on the support packages. The extension to more complicated image processing or interaction is only a matter of further software development.

2.3 Functionality

The experimental equipment has proven comfortable and easy to handle. Its speed is limited by the capability of the Vax computer, but all aspects of the visual servo problem may in principle be demonstrated. Equipment which simplify the implementation of an image laboratory have been used. The system has also successfully been used in experiments on inspection of GaAs-wafers (Silverberg et al, 1985), on motion detection in image sequences (Nielsen, 1984), and on measurement on the ash-line in bark ovens (Dahl and Nielsen, 1986). The latter experiments were performed using the video tape recorder to bring in images from an industrial bark oven.

3. THE SCENARIO

The Turtle will simulate an AGV working on the floor in a warehouse. The video camera is placed to get an overview of the scene, and is fixed in position. The Turtle moves on the floor performing a workcycle, which consists of visits to work stations. A human operator is assumed to define the workcycle and then not to be active or even present during the repeated cycles. The stations are the fundamental sites in the workspace symbolizing places where a load is collected or delivered, or where a task is performed. The requirements on the motion paths between stations are more flexible. The primary concern is that the Turtle reaches the stations. The scenario also includes obstacles, either described by the operator using the man-robot interface or detected by the program. The program checks that the motion paths of the Turtle avoid the known obstacles. When the Turtle detects new obstacles during operation the paths are replanned automatically. If the Turtle is unable to reach a station, due to new obstacles, it calls for help by sending an alarm to the operator.

The visual servo is provided with the capability to handle three different types of events when working unsupervised. Firstly, the Turtle can interact with the Ilon car symbolizing another (bigger) AGV which occasionally delivers material. The position of the Ilon car is not specified from time to time, but it is assumed to signal when it comes or leaves. The Turtle should go to the Ilon car and make a symbolic load transfer. Secondly, objects may unexpectedly appear in the work space. For example, humans may enter a risk zone. An external motion detector signals if something is moving into the scene. Thirdly, detected collisions must be handled.

3.1 The Workspace

A formal description of the scenario reveals the structure of the experiment. The notion of workspace is adopted (Brady et al, 1982). The extent of the workspace is here all points the robot can reach and the camera can see. The elements in the workspace are easily abstracted to geometrical objects in the floor plane. The elements are obstacles, stations, event points, and paths, whereas the geometrical objects are points, curves, and areas within closed curves. The curves are restricted to be polygons in the actual implementation. The workspace elements are defined as follows.

Obstacle: An obstacle is an area on the floor within a closed curve. It is a forbidden area for the robot. The obstacles entered by the operator are assumed to be stationary.

Obstacle map: The set of all known obstacles is called the obstacle map.

Station: Point on the floor. The stations are assumed to be fixed in position. The set of stations are ordered in sequence. The robot should visit the stations in the so defined order.

Event point: Point on the floor, which the robot has to visit to handle an event.

Path: A path is a curve (polygon) between two points on the floor. There are two types of paths. The path between two stations is called a station-to-station path, and the path to or from an event point is called a temporary path.

Workcycle: A workcycle is a set of paths forming a closed route, which the robot should run repeatedly.

Path attribute: The boolean path attributes are cooperate, verify, and explore. The attributes are associated with the station-to-station paths. They are used to determine whether or not an action may be taken in the event handling as described in Section 4. Typically the robot should postpone excursions when carrying a heavy load, and wait until the load is delivered.

4. THE ACTIONS

The visual servo is a system where the human operator is relieved of continuous supervision, but still should be available in emergencies. The needed robot actions will be presented starting with the primitive actions, then more complex actions, and finally the event handling.

Four motion procedures, similar to the Turtle procedure notation (Abelson and diSessa, 1980, Appendix A), are used to command the motion: Forw(length), Backw(length) TurnLeft(angle), TurnRight(angle). The length and angle are converted to the time the on/off signals of the Turtle motors have to be set to obtain a move. We will here use: Turn(angle) instead of TurnLeft(angle) or TurnRight(angle) to simplify the writing. The motion of the

Turtle to a desired position from its actual position is performed by first turning the desired angle and thereafter moving forward the desired length. The robot moves backwards only for fine adjustment of its position and after a collision. The state of the control is the last measured position and heading of the Turtle. We have

MoveTo(position) is
  Calculate(in position; out length,angle)
  Turn(angle)
  Forw(length)

The uncertainties in the Turtle position are vectors but we will work only with the magnitude. The top of the Turtle may vary the distance $r < 2cm$ depending on if the Turtle leans on the forward or on the backward support. The disturbances during normal motion are mainly due to varying friction between the wheels and the floor and to backlash in the drive train. The maximum error in position R after a MoveTo command is the sum of the static error r and the motion errors, and a rough estimate for a 15 cm MoveTo command is $R < 5cm$. The precision is thus poor, but the advantage of feedback is that crude systems can be used. Larger unusual disturbances may occur if the Turtle touches an obstacle or one wheel slips, for example on a piece of paper.

4.1 Basic actions

Three basic actions are sufficient to build the needed behavior of the robot. They are two motions: path following with image feedback (PathFollowing), contour following with the help of touch-sensors (ContourFollowingByTouch), and one planning action: path finding using a map (PathFinding).

PathFollowing. Image feedback is used to control the Turtle while following a path. The position of the marking symbol at the top of Turtle is determined in the image. This position is transformed to the position on the floor. The paths are restricted to polygons, where each segment is restricted to a maximal length of 15 cm. The strategy for following of the path is illustrated by an example in Fig. 4.1. The path is described by S - starting point, T - terminal point, V1-V4 - points of the path. The points P1-P8 show an example of a motion. The basic idea is to execute MoveTo commands to each of the polygon vertices. The state of the control is updated after each MoveTo by measuring in the image. A move is accepted if it is within the limits of normal motion disturbances R, which is illustrated as circles around V1-V4. If the robot is outside a circle then a large error has occurred and a new MoveTo command to the same point is performed, as for P1 and P2. At the terminal point we require the precision of the static error r. The Turtle moved backward from P7 to P8 to correct its final position.

ContourFollowingByTouch. The Turtle can follow an object contour by use of the touch sensors. The Turtle cannot move with its dome in constant contact with an object. Instead the touch sensing is based on repeated collisions as in Winston (1977, p. 248). After a collision the Turtle first moves the distance l away from the obstacle. Thereafter the path is a square with side $4l$. We use $l = 15$ cm. The robot will collide with the obstacle again while following the square path. The basic search motion is then repeated

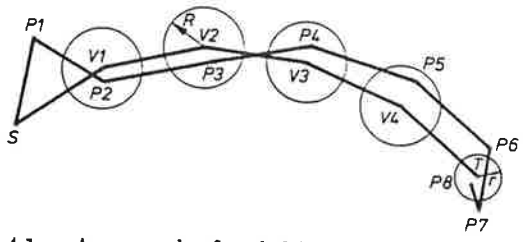

Fig. 4.1 An example of path following.

until either of two cases for a stop condition is fulfilled. In the first case the contour following is continued until the object is encircled. In the second case the contour following is continued until the robot reaches the planned path on the other side of the object. An example of a result of the contour following is seen in Fig. 4.2.

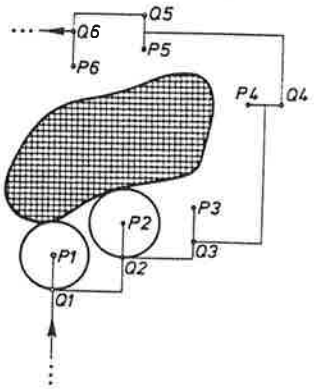

Fig. 4.2 An example of contour following.

The knowledge of the object contour is the points Pi and Qi which are measured from the image. The points Pi are the Turtle positions at the obstacle boundary after a collision. The points represent the center and are hence one Turtle radius away from the obstacle. The points Qi are the position after moving back $l = 15$ cm. Each line segment PiPi+1 is moved the distance of one Turtle radius to shrink the polygon. The shrunk polygon form the geometric description of the object. In the case when the search stops when the path is reached, then only an open part of the object contour is obtained. In that case the contour segment is extended to a closed curve, by introducing two extra points behind the end points P1 and PN of the open contour.

Pathfinding. A path finding algorithm is needed both for flexible operation and for task level programming (Brady et al, 1982). The path finding problem is simple to state: Given an obstacle map, an initial state and a final state of the robot; find a robot path that avoids the obstacles. The general and complete solution seems to be very complex (Brady et al, 1982). The visual servo uses image feedback, so we have the restriction that the robot must stay in the area which can be supervised by the camera. The restriction is described by introducing the mapping of the image boundaries onto the floor as obstacles in the obstacle map. A reason for a simple algorithm is that the motion disturbances can cause numerous collisions on a path of minimal length which worms between the obstacles.

Here a very simple approach is taken to the path finding problem. This is legitimized by the possibility to send an alarm to the operator. The algorithm consists of two steps. First a path described by a two vertex polygon is searched. The position of the two vertices are varied in a simple search, and after each modification of the path it is tested with the use of the obstacle map if the path is avoiding the known obstacles. When a free path consisting of a polygon with two vertices has been found the program attempts to smoothen the path by the calculation of a spline and then samples it to a many vertex polygon. It is checked to determine if the resulting path is free. If not then it is searched for a new two-vertex polygon. This procedure is repeated. If a smoothed path is not found then the first free two-vertex polygon will be used. The approach works in several simple cases. It is worth mentioning that in a first try to find an algorithm a one vertex polygon was used as a path, but the problem was to find paths that stayed within the image.

4.2 Event handling actions

PerformWorkcycle. The normal mode of operation is Path-Following on the paths of the workcycle.

ObstacleAvoidance. A collision may be with a new unknown obstacle or it may be with a known obstacle when the robot due to motion disturbances deviates from its path. Hence, the Turtle position is determined from the image and it is compared to the obstacle map. The Turtle returns the shortest way to its path if it was a known obstacle.

In a collision with a new obstacle the ContourFollowing-ByTouch algorithm is used. The Turtle has to decide on starting the contour following to the left or to the right. The decision is not crucial for the ability to find a path, because if one search direction is unsuccessful then the other search direction is automatically tried. However a good guess speeds up the contour search. The guess is based on the obstacle map, the planned path, and the state of robot motion. The search direction is chosen to be away from known obstacles. If there are no known obstacles influencing the path, then the direction of contour of the new obstacle is estimated, and the search direction is chosen to be the direction in which the robot moves forward. An obstacle contour is obtained from the contour following algorithm and the obstacle map is updated. The path finding algorithm is called to replan the paths of the workcycle to avoid the new obstacle.

VerifyAction and ExploreAction. An external detector can indicate a new object moving into the scene. The moving-object detector sends the detected object position, which is the event point for this event. There are two possible investigation actions.

VerifyAction: The path finding algorithm is called to plan a path to the event point, and the Turtle goes there. The Turtle verifies the presence of an object by the use of its touch sensors. The information is used e.g. to distinguish between objects and shadows. The path finding algorithm is called again to plan a path to the next station, and the Turtle goes there.

ExploreAction: As in the verify action the Turtle goes to the detected object position. If an object is sensed, then the contour following algorithm is called. The Turtle will as a result of this encircle the object, and an obstacle contour is obtained from the contour following algorithm. The obstacle map is updated and the path finding algorithm is called if any path in the workspace needs replanning. The path finding algorithm is called again to plan a path to the next station, and the Turtle goes there.

RobotCooperation. The following sequence of actions is performed in the case of robot cooperation. The Ilon car is used. The marking symbols of the Ilon car are known. Both the position and the orientation of the Ilon car is determined from the image, and the area on the floor occupied by the car is calculated. The event point 10 cm in front of the Ilon car is also calculated. The obstacle map is temporarily updated with the Ilon car as an obstacle. The path finding algorithm is called to plan a path to the event point. The Turtle goes there using the path following algorithm. The Turtle then starts a rendez-vous operation by moving straight to the Ilon car until the touch sensors verify contact. The Turtle then returns to the event point. This is a simulation of e.g. a load transfer. The path finding algorithm is called again to plan a path to the next station. The Ilon car signals when it leaves and the temporary description of the Ilon car as an obstacle is then removed from the obstacle map. The Turtle continues the workcycle.

4.3 The event monitor

An event monitor controls the robot actions. Its purpose is to give the robot a path to follow. The information available to the monitor is the geometric description of the workspace, the state of the robot, and the event signals. The monitor introduces the event points and the temporary paths, and may introduce obstacles in the obstacle map. It replans the station-to-station paths according to the updated obstacle map. The stations, the path attributes, and the obstacles defined by the operator are never changed. The control structure resembles an operating system of a computer, and an outline of the event monitor with possible events, path attributes, event points, and actions is described as follows.

```
EventMonitor is
    Background job:
        Action:     PerformWorkcycle
    Collision event (First priority):
        Action:     ObstacleAvoidance
    Object detection event (Second priority):
        Path attributes: verify and explore
        Event point: detected object position
        Action:     if verify and explore then
                        ExploreAction
                    else if verify then
                        VerifyAction
    Robot cooperation event (Third priority):
        Path attribute: cooperate
        Event point: 10 cm in front of the other robot
        Action:     if cooperate then
                        RobotCooperation
```

An event signal requires a check of the path attribute values on the present path. If the event is not handled immedi-

ately then the event is put in a list of events yet untreated. The event list is scanned by the event monitor each time the robot is at a station. It is checked to determine if the path attribute values for the next station- to- station path allows handling of any event in the list. An event will not be treated if there is no path in the workcycle with attribute values that allows handling it. There is an alarm possibility at all levels if an action cannot be completed. If it is possible to continue the robot only notifies the operator. Otherwise it calls for help and waits until the operator intervenes.

5. THE MAN-ROBOT INTERFACE

A man-robot interface has been designed to provide two main services. One is the programming where the the workspace and tasks are defined. The other is supervision facilities at different levels of detail.

The man-robot interface is used to describe the workspace i.e. to enter stations, obstacles, paths, and path attributes. Hence the facilities of this interface is concentrated on motion. Other important aspects such as gripping and manipulation are not treated. Different approaches to robot motion programming have been identified (Craig, 1986). In explicit programming, the user specifies all of the motions needed to accomplish a desired path by giving an explicit list of coordinates. Programming by teaching is done by guiding the robot manually and storing the path. The man-robot interface uses a manual mode where the robot is controlled using the joysticks. The highest level of programming is task level programming. The user specifies geometric models and descriptions of tasks in terms of these models. The detailed motions are derived automatically from these specifications (Brady et al, 1982).

Correspondence between image and workspace. The mapping between the floor plane and the image is one to one. There is thus a unique correspondence between the geometrical objects of the workspace and points and curves in the image. The geometric description of the workspace may be entered by pointing in the image using a graphical editor.

The graphical editor. The key idea in this work on robot programming is to explicitly use interaction based on real gray scale images of the scene. These images are presented on the monitor screen. A graphical editor allowing color graphical manipulations on these images has been developed. The terminal, the mouse, the joysticks and the monitor screen (Fig. 2.1) are used for the interaction. The mouse is used for pointing in the image on the monitor screen. The commands to the editor are selected from menus on the terminal. The output of the graphical editor is overlayed on the gray level images.

The editor can handle points and curves. Two types of curves are possible: polygons and cubic B-splines (Newman and Sproull, 1979, Section 21.4). The shape of a polygon is defined from its vertices. The shape of a cubic B-spline is defined from control points. The editor can be used to interactively enter, copy, move, rotate, or delete both points and curves. The shape of a curve is edited by adding, moving, or deleting the shape defining points (a polygon vertex or a control point of a cubic B-spline). The technique with cubic B-splines is well established. The control points have local support. This means that if a control point is moved then the curve is changed only in the neighborhood of that point. It also satisfies the requirements of shape stability in editing, which means that the shape of a curve is not drastically changed if a control point is moved slightly (Newman and Sproull, 1979, Chapter 21). These properties makes them feasible for trajectory generation.

Path programming. The use of graphics in images of the workspace simplifies the motion trajectory generation. A spline is entered using the graphical editor. The spline is automatically transformed to a Turtle path by sampling it to a polygon, which when projected on the floor has no segment longer than 15 cm. The program of course checks that the obtained path avoids the obstacles. The result is a smooth polygon path. In other current systems for trajectory generation several intermediate positions, via points, have to be defined to obtain a smooth path (Craig, 1986). Here only a few control points have to be defined. It is easy done by pointing in the image and the result is directly presented.

How to enter the workspace. Commands are used to determine whether the graphical input is obstacles, stations or paths. The coordinates in the image are transformed to points on the floor. The obstacles, stations, paths and path attributes are entered in the following way.

Obstacle: Entered as a closed polygon. The vertices are pointed out directly in the image. Presentation: Red polygon filled with a white grid net.

Station: A station may be entered in three ways: 1. Point in the image with the mouse. 2. Control the robot to the position of a station using the joysticks, and give a command which tells that this is a station. The robot position is determined from the image. 3. Tell the robot that a certain feature in the image is a station. The only features accepted as a station definition are a specified set of marking symbols. Presentation: Blue dot.

Path: Entered as a number of points, which by command are interpreted as a polygon or a cubic B-spline (default). Both the spline and the sampled Turtle path are presented during the editing, and thereafter only the Turtle path is presented. Presentation: Polygon in color according to the values of the path attributes.

Path attributes: Entered by placing the cursor on the path and then give the desired path attribute values by commands. A color code is used to indicate the different combinations for the attribute values.

Task level programming. When a new station is entered a path avoiding the obstacles is calculated by default, using the path finding algorithm, and presented to the operator. Hence, task level programming is done here using graphics. The operator may of course modify the calculated path using the path programming.

Presentation of Status. During operation there are three levels of detail for the presentation on the monitor screen. Firstly, the monitor can just display the working scene as seen by the camera. Secondly, the graphical description of the workspace can be overlayed the image. Thirdly, the steps of the image processing can be added to the presentation, giving a complete record of the internal status

and algorithms. The actual processed part of the image is marked, the output of the edge detector is displayed, etc.

6. THE IMAGE INTERPRETATION

The image interpretation in the servo will be presented. First, one approach to image processing representing the state of the art for industrial vision systems will be reviewed. Based on this approach the design of marking symbols, that are described by invariants under parallel projection will be treated. The derivation of invariants, the symbols, and the algorithms for recognition of the symbols are given. The image interpretation uses a number of decision thresholds that are assigned values based on an error analysis.

6.1 An Approach to Image Processing

Special purpose parallel hardware that can identify an object from a set of a hundred objects in less than a second is available (Petersson, 1983). The input is a multilevel grayscale image. A gradient operator is applied to the image to extract the contours (Rosenfeld and Kak, 1982, Section 10.2). The gradient values are iteratively refined using image parallel relaxation (Rosenfeld and Kak, 1982, Section 10.5). The refined gradient values are thresholded to obtain a bilevel image representing the contours. The contours in the bilevel image are described with a labeled tree for each object, see Fig. 6.1. An object is defined by an outer closed contour and the contours contained within it. Going down in the tree means being inside. Each contour (each node in the tree) is labeled with its properties. Firstly, integrated measures of the contours are used. Each curve is represented by its position C (centroid) relative to the position (centroid) of its enclosing contour, by its area A, and by its perimeter P. Secondly, different shape attributes like the number of corners N may be extracted (Pavlidis, 1980). The labeled tree is sufficient for identification of objects in applications with a camera mounted above a conveyor belt. The objects may arrive on the conveyor belt in an arbitrary position and orientation within the plane of the conveyor belt. The labels C, A, P, and N are, however, invariant under two-dimensional rotation and translation. So, the tree can be stored for each object in a set, and the recognition of a specific object is done by comparing its tree with the set of trees.

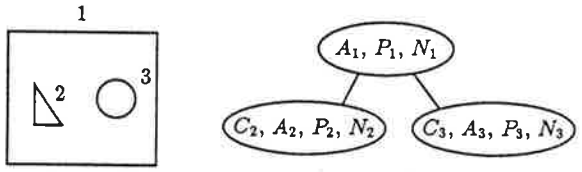

Fig. 6.1 Object representation by a labeled tree representing the contours and their properties.

The situation changes in an application where the objects can be arbitrarily oriented in three dimensions relative to the camera. The shape and size of contours vary. The number of poses is infinite and it is impossible to store all the cases for comparison. The number of corners is invariant, but it may be hard to detect. A sharp corner may be projected to a wide angle corner. The solution is to use invariants under three-dimensional projection.

6.2 Invariants Under Parallel Projection

Imaging can be described by parallel projection if beams from different points of the object to the corresponding image points can be considered parallel. Parallel projection has the properties that parallel lines map to parallel lines, and that the ratio of division k between any three points on a straight line is preserved. Invariants based on integrated measures are easy to derive from these properties. Consider objects consisting of two concentric similar curves as in Fig. 6.2. Let the inner curve be parameterized as $r_1(\theta)$, and the outer curve be given by $r_2(\theta) = kr_1(\theta)$ where the ratio of division k is a constant.

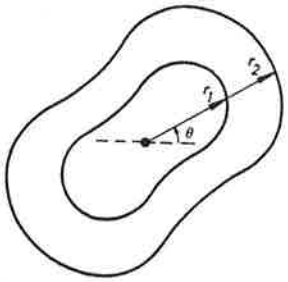

Fig. 6.2 Two concentric similar curves.

The area and perimeter inside the closed contours $r_j(\theta)$ in Fig. 6.2 are

$$A_j = \frac{1}{2}\int_0^{2\pi} r_j(\theta)^2\, d\theta \qquad j = 1,2$$

$$P_j = \int_0^{2\pi} \sqrt{r_j(\theta)^2 + (\frac{dr_j(\theta)}{d\theta})^2}\, d\theta \qquad j = 1,2$$

Invariants based on integrated measures follow:

$$\frac{A_2}{A_1} = k^2 \qquad \text{and} \qquad \frac{P_2}{P_1} = k$$

The invariants classify uniquely for $k > 1$, and a continuum of separate objects can be obtained by varying the ratio of division k. The results hold in particular for two concentric squares or two concentric triangles.

6.3 Objects and Recognition

Objects. A set of six marking symbols is used. They are based on the invariants presented in the previous section. The set of marking symbols consists of three versions of two concentric squares ($k = 1.57, 2.20, 3.23$), and three versions of two concentric triangles ($k = 1.58, 2.00, 3.21$). The outer dimension of the symbols are approximately the same. The reason to use triangles and squares is that their shape can be verified, for example using the shape classifier presented below. The Turtle is marked with one symbol. The Ilon car is marked with two symbols, see Fig. 2.2. One symbol is used as a reference in the start up procedure. This symbol and the two remaining symbols in the set may be used to define stations as discussed in Section 5.

Recognition. The marking symbols are black and white. Therefore a simple contour extraction suffices. The Roberts

operator is used to compute gradients on the grey-level image $f(x,y)$ (Rosenfeld and Kak, 1982, Section 10.2). The magnitude of the gradient is computed as $g(x,y) = max(|f(x+1,y+1)-f(x,y)|, |f(x,y+1)-f(x+1,y)|)$. A bilevel image $B(x,y)$ representing the contours is obtained by thresholding $g(x,y)$ with a threshold t_b. The nominal value of t_b is 40, which should be related to the image dynamic 0-255. The connected components are extracted from the bilevel image B(x,y), using the 8-connectedness border following algorithm and the border finding algorithm in Rosenfeld and Kak (1982, Section 11.2). The labeled trees are then built. A crude screening is made first where small contours are disregarded. The tree representing the inside/outside relations is simple to obtain, and so are the integrated measures C, A, and P.

The curvature changes under projection and corners may be hard to detect. Here a special shape classifier has been developed that can classify a contour (in fact any set of points) γ as a triangle, a quadrangle or neither. It performs successfully even if a corner is projected to a wide angle. A two step algorithm is used. The first step leads to an hypothesis about the number of corners together with rough estimates of their positions. The centroid r_c of γ and the point r_1 in γ which is most distant from r_c are computed first. An orthogonal coordinate-system is defined as in Fig. 6.3. The x-axis goes through r_c and r_1, and r_1 is chosen as origin. The extremum-values of γ in this coordinate-system are then computed, and we set $r_2 = y_{min}$, $r_3 = x_{max}$, and $r_4 = y_{max}$. Finally the point r_5 is the intersection between the x-axis and the line through r_2 and r_4. We have $|r3 - r1| = |r5 - r1|$ for a triangle and $|r3 - r1| = 2 \cdot |r5 - r1|$ for a parallelogram. A hypothesis N_H about the number of corners is now formed as $N_H = 3$ if $|r_3 - r_1| < 1.5 \cdot |r_5 - r_1|$ and $N_H = 4$ otherwise. A first estimate of the corner positions is given by r_1, r_2, r_4 if $N_H = 3$ and r_1, r_2, r_3, r_4 if $N_H = 4$. The second step uses the corner estimates to divide γ into N_H subsets γ_j corresponding to the different sides of the object. Straight lines l_j are then fitted to each γ_j using linear regression. Different tests can be used to determine whether γ_j can be considered as a straight line. We require that all points of γ_j be within the distance t_l to l_j, for γ to be considered as triangle or quadrangle. If it is satisfied we set $N = N_H$, and if it is not satisfied the contour is classified as neither triangle nor quadrangle. The corners are more accurately determined as the intersection of the lines l_j in the case $N = N_H$.

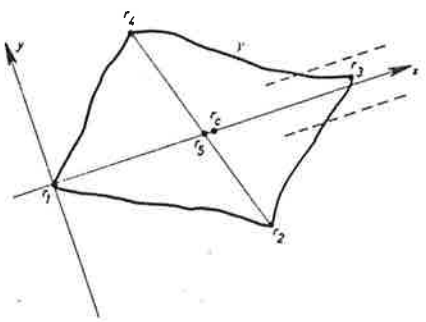

Fig. 6.3 The local coordinate system and the points of γ extracted by the shape classifier.

The symbols are composed of either two triangles or two quadrangles with the same center. Let the subscript 1 and 2 denote the outer and inner contour respectively. The labeled tree matches a known symbol denoted by the subscript 0 if

$$N_1 = N_2 = N_0$$
$$|C_1 - C_2| < t_C$$
$$|\frac{A_2}{A_1} - k_0^2|\frac{1}{k_0^2} < t_A$$
$$|\frac{P_2}{P_1} - k_0|\frac{1}{k_0} < t_P$$

The selection of the thresholds is discussed in combination with the error analysis.

Subimages. All image data is not always processed. Only a rectangular subimage is interpreted when the Turtle is tracked during path following. The predicted position of the marking symbol of the Turtle is used as the midpoint of the subimage. The size of the subimage is determined by a length scale S, which depends on the predicted distance r from the camera as $S = S_0 r_0 / r$. The reference length scale S_0 of the subimage is chosen as the sum of the symbol image size and the effect of robot motion errors both evaluated at a reference distance r_0.

6.4 Analysis and Decision Thresholds

There are many imperfections in the image data. The two main errors in γ compared to an ideal image of the Turtle symbol are rounding of corners and a bias towards brighter areas in edge position. Rounding of corners means that the connection between two sides is a smooth curve rather than a sharp vertex. This problem can be neglected because the second step of the shape classifier compensates by fitting straight lines. Errors of perspective are negligible (Nielsen, 1985).

Estimates of maximal errors are used for the selection of the four decision thresholds t_l, t_C, t_A, and t_P. This means it has been chosen to be tolerant on each single test. The combination of tests is still safe. The contour segments γ_j are normally fairly close to straight lines. The rounding of corners may, however, cause a few points at the end to give larger deviations. The threshold t_l which governs the maximal allowed deviation is chosen so that no point of a contour segment is allowed to be more than 3 pixels away from the line. The position of the centroids depend on discretizing effects, and t_C is set to correspond to half the diagonal of the inner symbol.

Bias. Both the perimeter and the area of the inner quadrangle will be overestimated because of the bias in the edge detection. The outer quadrangle on the other hand is underestimated. Estimates of the maximal relative errors are obtained for the Turtle symbol as follows. It has been experimentally verified that the values obtained are suitable both for the Turtle symbol and the other symbols used. Referring to Fig. 6.4, assume parallel projection, let the inner square be mapped to a parallelogram with sides b_1 and b_2, and denote the bias with e. The bias changes the sides of the parallelogram by the distance e. The area- and perimeter-quotients are then

$$\frac{A_2}{A_1} = \frac{(kb_1 - 2e)(kb_2 - 2e)}{(b_1 + 2e)(b_2 + 2e)}$$

$$\frac{P_2}{P_1} = \frac{(kb_1 - 2e + kb_2 - 2e)}{(b_1 + 2e + b_2 + 2e)}$$

The expressions degenerate to the invariants for e = 0. Let pixel size be the length unit. In the experiment we have roughly $e < 1$ and $b_j > 20$. Then

$$|\frac{A_2}{A_1} - k^2|\frac{1}{k^2} < 0.25 \text{ and } |\frac{P_2}{P_1} - k|\frac{1}{k} < 0.13$$

Hence we use the values $t_A = 0.25$ and $t_P = 0.13$. Note that the relative error in the perimeter quotient is smaller than than the error in the area quotient. We form the ratio between the relative errors to investigate this. The minimum value with respect to b_1 and b_2 is obtained for $b_1 = b_2 = b$. Now introduce $x = e/b$ giving $|x| < 0.05$. Then

$$\frac{Rel.error\ in\ A_2/A_1}{Rel.error\ in\ P_2/P_1} = 2\frac{1 + x(k-1)/k}{1 + 2x} > 1.87$$

This expression explains an experimental finding. The errors in the perimeter-quotient were always smaller than the errors in the area-quotient. This is the normal situation if the main error is a bias in the position of the contour.

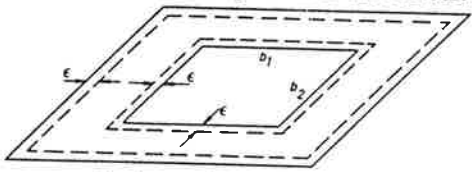

Fig. 6.4 The Turtle symbol under parallel projection. The broken lines illustrate the effect of the bias in the edge detection.

Iteration. The image interpretation steps are normally performed in sequence, but it is iterated if the identification of the symbols fails. Then it is assumed that identification fails either if the contour is not properly extracted or if the symbol is not within the subimage. The latter may occur due to large motion disturbances. A contour can be broken due to image noise. The other parts of the image interpretation are robust under the experimental conditions described here. The threshold of edge detection t_b and the size of the subimage S are alternately modified. The size S is doubled in the first iteration but t_b is kept as 40. The next two iterations use the same S but t_b is reduced to 35 and 30 in sequence to get more contour points. If the interpretation fails then the whole image is processed with $t_b = 40$. The values $t_b = 35$ and 30 are then tried. An alarm is sent if the symbol is not found during these iterations.

7. EXPERIMENTAL DATA

The dimensions of a typical experiment are presented in Fig. 7.1, in meters. The angle between the optical axis of the camera and floor is 32 degrees. The camera position is in the coordinate system (-0.45, -2.45) and the height above the floor is 1.85 m. The part of the floor seen by the camera in the actual experiment is marked in the map. There are three stations numbered 1, 2, and 3. There are also three obstacles A, B, and C in the scene. The obstacles are a chair A, a box B, and a variable-voltage transformer C between the stations two and three. The obstacles A and B symbolize permanent effects, for example machines, which are defined as obstacles using the man-robot interface. They are thus known to the robot. The obstacle C will purposely not be defined. It represents an unknown obstacle like something dropped by another AGV. The AGV will collide with it on the way from Station 2 to Station 3.

The AGV successfully performs its tasks. An automatic start up procedure calculates actual perspective and determines Turtle position and heading. A normal station to station travel takes less than a minute. The obstacle C is explored, the obstacle description is included in the obstacle map, and the path is replanned. The AGV also performs a rendez-vous with the Ilon car. The start-up and the completion of the workcycles are performed completely automatically by the robot without any human guidance or help. In (Nielsen, 1985) a sequence of 36 images is used to illustrate the behavior.

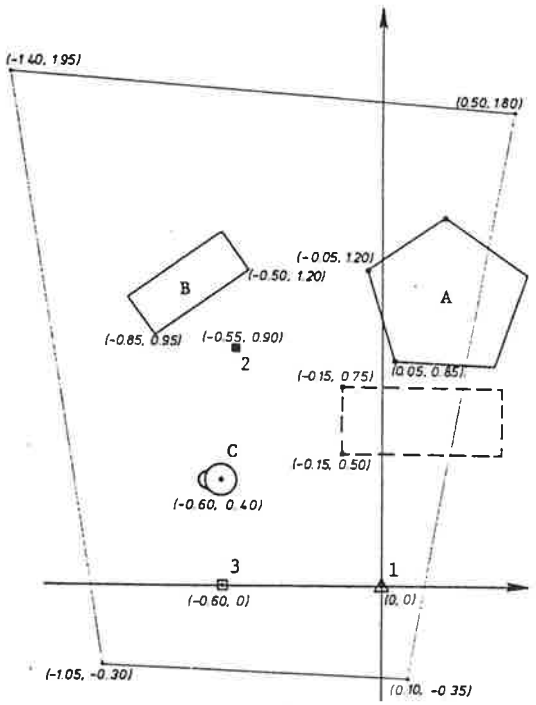

Fig. 7.1 A map of the floor showing the spatial dimensions of the experiment. The map is generated by the system.

The implementation. The hardware described in Section 2 has remained unchanged but the software has developed in different directions during the experiments. Parts of the experiment have been presented earlier (Nielsen and Johansson, 1984). The code consists of 13987 lines of Pascal representing an amount of 387 kbytes of source code. No special attempts have been made to obtain compact code. The executable code size is 146 kbytes. The source code is distributed in the following way.

Support packages	10 %
Turtle control	27 %
Image interpretation	21 %
Man-robot interface	42 %

The support packages include the packages mentioned in Section 2, in addition to a package for handling lists and a package for handling of splines. The image interpretation code size includes the start-up estimations.

8. CONCLUSIONS

There is not yet a problem formulation of visual servoing which captures the essential properties and admits an analytical or numerical solution. It is thus essential to build a system, and try significant experiments. In the work presented in this paper a robot moving on the floor is controlled by image feedback. An abstraction of the workspace reveals a simple geometric structure. The actions needed to obtain flexible functioning of the robot can be composed from three basic actions: path following with image feedback control, obstacle avoidance by touch, and path finding using a map. If real-time image processing hardware is used, the servo presented here illustrates how an an industrially useful behavior of an AGV could be obtained.

A flexible system increases the requirements on good interaction with an operator. The man-robot interface is based on an operator manipulating interactive color graphics overlaid on images of the working scene of the robot. The interaction is accomplished by specifying obstacles and stations rather than explicit path coordinates. The graphics are entered simply by pointing in the image, since the mapping between floor and image is one to one. In this way it is feasible to use a new path programming technique requiring few control points instead of current methods involving several via points. Furthermore, task level programming can be done by pointing out robot destinations in the image. The use of graphics here is thus very different from other recent uses in robotics, where graphics have been used for simulation but not to simplify the real-time interaction (Craig, 1986).

Visual recognition in the three-dimensional scene has to cope with the problem of objects having infinitely many poses. Obviously all poses cannot be stored, and therefore one must rely on more dense descriptions. A key contribution in this work is the design of marking symbols, which are described by invariants under parallel projection. The status of image processing hardware has to be considered in the design. The marking symbols designed here are black and white, which simplifies edge detection and contour description. The recognition problem is thus converted to a problem of designing symbols and algorithms to detect the symbols. One example of a new algorithm is the shape classifier. It was developed to solve the problems caused by curvatures changing under projection. The experiments have shown it to be simple and reliable. The invariants used here are based on integrated measures and thus are not sensitive to individual pixel errors. The invariance of the area-quotient is well known. It seems to be a new, though trivial, observation that the perimeter-quotient is also invariant. Nevertheless it is tractable, since the perimeter-quotient is the more precise criterion under the typical errors. The problem formulation, the results, and the analysis are believed to be new contributions.

9. ACKNOWLEDGMENT

This work has been supported by the Swedish Board for Technological Development (STU-82-3429).

10. REFERENCES

Abelson, H. and A.A. diSessa (1981). *Turtle Geometry. The Computer as a Medium for Exploring Mathematics.* The MIT Press, Cambridge, Mass.

Brady, M., J.M. Hollerbach, T.L. Johnson, T. Lozano-Perez and M.T. Mason (Eds.) (1982). *Robot Motion Planning and Control.* The MIT Press, Cambridge, Mass.

Craig J. (1986). *Introduction to Robotics.* Addison-Wesley.

Dahl, O. and L. Nielsen (1986). Ash-line control. Same conference.

Newman, W.M. and R. F. Sproull (1978). *Principles of Interactive Computer Graphics.* McGraw-Hill.

Nielsen, L. (1984). Motion detection in image sequences. *Preprint, NSF-STU International Workshop on Computer Vision and Industrial Applications*, May 14-18, Stockholm, Sweden.

Nielsen, L. (1985). *Simplifications in Visual Servoing.* Ph D thesis CODEN: LUTFD2/ (TFRT-1027), Department of Automatic Control, Lund Institute of Technology, Sweden.

Nielsen, L. and K. Johansson (1984). Robot med egna ögon. *Industriell Datateknik*, 1984:5, pp. 25-29.

Pavlidis, T. (1980). Algorithms for shape analysis of contours and waveforms. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. PAMI-2, no. 4, pp. 301-312.

Petersson, C.U. (1983). An integrated robot vision system for industrial use. *Rovisec-3, Third International Conference on Robot Vision and Sensory Control*, Nov. 6-10, Cambridge, Mass.

Rosenfeld, A. and A.C. Kak (1982). *Digital Picture Processing.* Academic Press, New York.

Silverberg, P., L. Nielsen, P. Omling, L. Samuelsson (1985). EL2-maps from computer based image analysis of semi-insulating GaAs wafers. *Symposium on Defect Recognition and Image Processing in III-V Compounds*, July 2-4, Montpellier, France.

Winston, P.H. (1977). *Artificial Intelligence.* Addison-Wesley.