CAR–TR–337
CS–TR–1962

DACA76–84–C–0004
December 1987

Production of Dense Range Images with the CVL Light-Stripe Range Scanner

Daniel DeMenthon, Tharakesh Siddalingaiah and
Larry S. Davis

Computer Vision Laboratory
Center for Automation Research
University of Maryland
College Park, MD 20742

ABSTRACT

This report describes a system able to produce 512 × 512 range images of model scenes in the laboratory. This ranging instrument, which comprises a light-emitting slit, a cylindrical lens, a step-motor controlled mirror and a CCD camera, is compact enough to be mounted on the tool plate of a robot arm. The light source itself is mounted away from this structure, and the light is brought to the slit by a flexible fiberoptic light guide. The robot arm's motion can be controlled by inputs from the range scanner, for simulation of autonomous vehicles equipped with rangers. This system is programmed to produce range images which are comparable in many respects to range images produced by laser range scanners. With this similitude of formats, software for edge detection, object recognition, dynamic path planning or data fusion with video images can be developed on range images produced by this laboratory equipment and can be easily ported to laser ranging systems.

The support of the Defense Advanced Research Projects Agency and the U.S. Army Engineering Topographic Laboratories under Contract DACA76–84–C–0004 (DARPA Order 5096) is gratefully acknowledged.

1. Introduction

This report describes hardware, interfaces and codes for the CVL light-stripe range scanner. It also includes instructions for the production of range images. The development of this ranger took place as a part of the Autonomous Land Vehicle (ALV) project. The vehicle tested by Martin Marietta in Denver carries a laser range scanner, and therefore the ALV project required the development of fast and robust algorithms for obstacle avoidance, path planning and object recognition from range images. We did some work directly with range images obtained from the vehicle, but we needed a more flexible system. Data collection with the vehicle is a complex and time-consuming operation. Our first alternative approach was to develop a software package, described elsewhere [1], able to produce synthetic range maps of scenes defined by Cartesian coordinates of significant points. However, with this type of input, only images of very simple scenes of planes, polyhedral and conic surfaces can be produced. For instance, synthesizing range images of bushes, rocks or vehicles would be unfeasible. On the other hand, models of these items can be purchased at model train shops or custom-built, and scanned by an appropriate system.The laser ranger used on the ALV is a bulky piece of equipment inappropriate for laboratory use; laser rangers for shorter ranges and suitable for laboratory use on models are under development [2] but were not available at the time of this writing. We found that a *light-stripe range scanner* would be suitable for obtaining range images for model scenes; furthermore, this type of scanner has more points in common with a laser ranger than is apparent at first, and the images obtained are very similar to laser ranger images. This similarity is described in Section 2. In particular, dense images can also be obtained; therefore algorithms performing low level processing at the pixel level in the range image prior to 3D reconstruction, such as edge detection, segmentation and edge-preserving smoothing, can be tested on range images obtained by a light-stripe scanner before they are applied to range images.

At this point we need to specify our use of a few terms specific to ranging techniques combining a camera and a projector. These techniques determine ranges by triangulation of the rays of light coming from the projector and rays of light reflected from the patterns created by the projector on the world scene. The generic term for ranging by light-pattern projection is *structured-light ranging*. In our specific construction, the projector creates a *light-sheet* or *plane of light*. In all generality, a sheet of light could be a very general ruled surface, but we imply here that the sheet is *flat*. A *sheet* can be allowed a certain thickness; therefore the term *light-sheet* is useful for discussions about the actual light output of the scanner, and also for considering the actual dispersion of the rays away from an ideal plane, which produces *light-stripes* of nonnegligible thickness when intersecting a world scene. The term *plane of light* will be preferred when referring to the ideally thin planar core of the light-sheet, which illuminates approximately the midpoints of the light stripe. Referring to this planar core is useful in geometric considerations and

angular measurements. The method of creating light-stripes on a scene with a light-sheet projector for ranging measurements is sometimes called *light-striping*. We can talk interchangeably of *light-sheet projector* or *light-stripe projector*, the light-stripe being a part and a consequence of the light-sheet. But obviously, when considering the camera image, we deal mainly with the *light-stripe*, the only visible part of the light-sheet in clear air. This visual prevalence of the light-stripe made us prefer the term *light-stripe range scanner* to *light-plane range scanner* for designating the whole system.

Range data acquisition using a light-stripe projector and a camera is a well-known method. For a general overview of this and other range-finding techniques, see Jarvis [3]. The projector and the camera are in a known mutual configuration. Ranges of world points illuminated by the light-sheet can be obtained by triangulation, since the distance from the light source to the camera nodal point is known, and the angles of the light-sheet and of the ray from the illuminated world point to its image can be easily estimated (see Section 13). Variants of this range-measurement method exist and differ in the method of sweeping the scene with the light-sheet :

• In industrial environments, the scene may consist of parts translated on a conveyer belt which have to be recognized. The conveyer belt motion provides the relative sweeping of the parts and the light-sheet.

• For systems mounted on a robot arm, the motion of the arm it self can be used to sweep the stripe [4],[5]. In this case, the camera and the stripe projector are displaced together. However, a robot arm is usually not an adequate tool for the small, fast and precise displacements which are required for scanning a light-stripe, because it is a heavy piece of machinery with a lot of inertia, and it makes variable errors according to which joint motions are involved in the displacement of the end plate [5].

• A third option has been chosen in our design. A mirror is rotated to project the light-sheet at various angles. This type of system incorporates a camera and a light-stripe scanning system that we call a *light-stripe range scanner*. Light-stripe range scanners are not new. Shirai [6] obtained range maps from a light-stripe scanner in 1972 and developed algorithms for recognition of polyhedral objects. Boyer and Kak [7] base their preference for a complex color-coded structured-light scheme on the argument that gear lash in the rotating mirror mechanism of a light-stripe range finder can contribute significantly to stripe positioning errors; but we feel that this problem can in fact be minimized to almost any desired degree by using a gearbox with the required (high) precision and by always rotating the gearbox in the same direction.

In our laboratory, the ranger is mounted on the tool plate of the robot arm (Figure 1). This does not mean that the robot arm is used for scanning. On the contrary, the robot arm has to be kept still during range image acquisition, while the mirror takes care of the scanning. There are two reasons for mounting the ranger on the robot arm. First, by controlling the robot arm by keyboard input of Cartesian coordinates or angles, it is relatively easy to position the ranger at any desired pose within the operation space of the

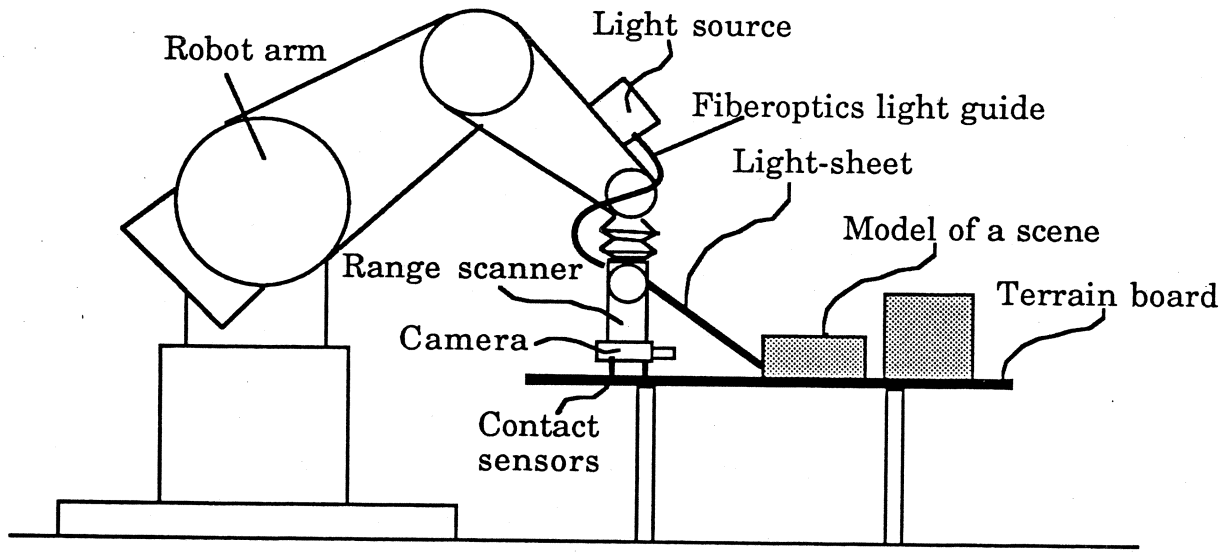

Figure 1: Configuration of the ranging system mounted on a robot arm for simulation of a ranger-equiped vehicle.

The robot arm itself is not used for light-sheet scanning. This task is performed by a stepper motor rotating a mirror. The robot arm is kept still during range data collection and displaces the range scanner between range images.

robot. Second, in a vehicle simulation mode, the displacement of the robot can be driven under software control; the robot motion is then controlled by software which uses range image analysis, in the same way that the trajectory of a vehicle would be controlled by software using laser ranger data [8].

One desirable feature which could not be attained is real-time range image acquisition. Our system uses a software stripe search on the digitized camera images and fills each range image pixel with a range value in about 2 milliseconds. This adds up to 8 seconds for a 64 x 64 image, and close to 8 minutes for a 512 x 512 image. Hardware solutions are being developed to divide these times by a factor of 10, by application of techniques comparable to those described by Ozeki et al. [9].

2. Dense Images from a Light-Stripe Scanner

In most of the light-striping techniques described in the literature, the grey levels from the pixels in stripe images from the camera are simply replaced by values coding the ranges to the corresponding world points. The end result after scanning the scene is a striped image of the scene, with the pixel values of the stripes changed to represent the ranges, and the other pixels between the stripes set to 0 (or 1). The problem with such images is that we would like to compress the lines together to eliminate these gaps between stripes, to obtain dense images to which templates can be applied for low-level image-processing operations. If the stripes were straight, they could be pushed one against the other. But the stripes *are* in fact straight when they are seen from the point of view of the mirror, since a stripe is then viewed within the plane of light which created it. The solution is therefore to obtain the ranges from the mirror, and build an image in which each row is assigned to the range data obtained for a given stripe. Thus the rows are indexed by the mirror steps. This solution also seems to have been chosen by Lozano-Perez et al.[10], for the purpose of doing edge detection on a range image with standard edge operators.

Notice that this is precisely the type of range image also obtained by a laser range scanner doing raster scanning with two mirrors. This parallelism between the two systems is illustrated in Figure 2. For every step of the mirror controlling the vertical scanning, the laser pulse is scanned (by the other mirror) in a plane containing the axis of the first mirror, and sweeps over a stripe of world points. The ranges of these world points are placed in the same row of the range image.

3. Architecture of the CVL ranging system

Figure 3 illustrates the architecture of the system. This figure shows the elements of the hardware, their interconnections, and also the elements of the software which control them and the image planes in computer memory where the range images can be found, for display and transfer to other computers.

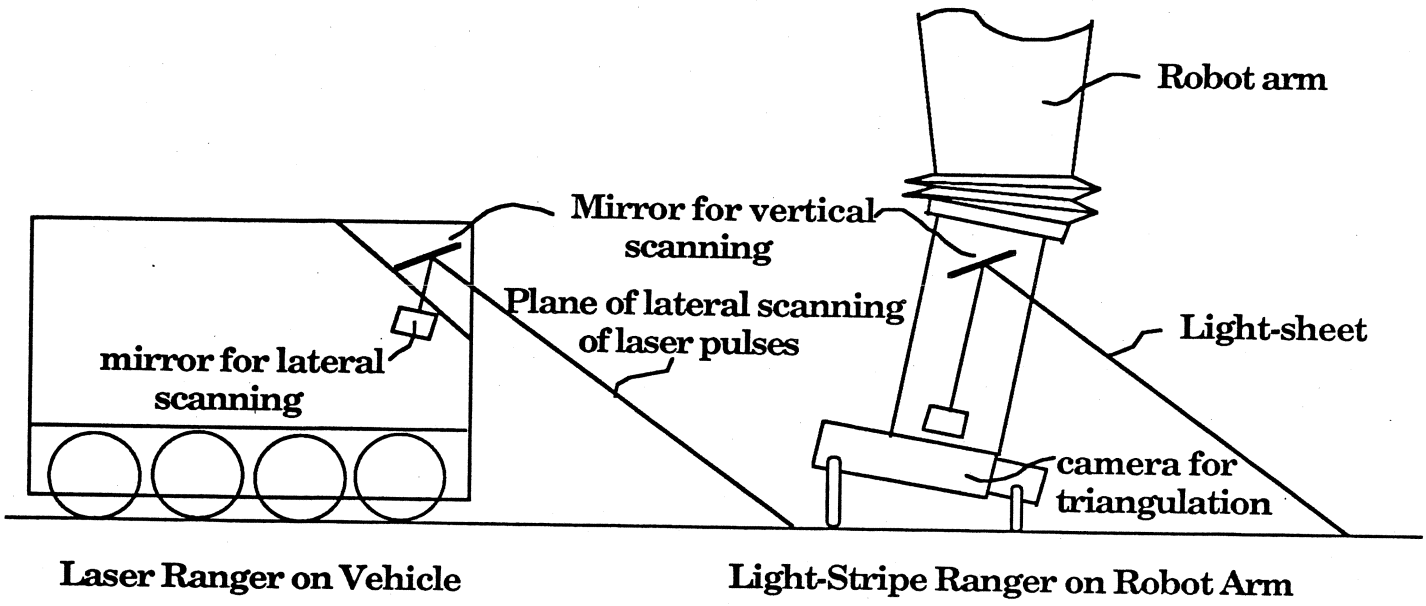

Figure 2: Light-Stripe Range Scanner as a simulator for laser rangers

In a laser ranger, the mirror with the horizontal axis controls vertical scanning. For any given step of this mirror, the pulse is scanned at constant angular increments within a plane which contains the horizontal axis of this mirror, and the ranges for points on the trajectory of the pulse are obtained.
These ranges fill a range image in which a new row is used for each step of the mirror.
Similarly, in the light-stripe ranger, a mirror controls vertical scanning. For any given step of this mirror, the ranges from the mirror to the world points which reflect the light are measured. For any given step of this mirror, these ranges are stored in a specific row of the range image.

The following sections of this report are a step-by-step description of the components of Figure 3. The order of presentation follows approximately the stream of information from the world points in a scene to the range pixels on the display screen, and the tools which transform this information are presented. The inverse transformation is described in Section 19. Finally, Section 20 gives directions for the production of range images with this system.

4. Description of the Camera-Scanner Block

The frame of the ranger is mounted on the end plate of a robot arm, and supports the camera and the stripe-scanning mechanism (Figures 1 and 2).

Figure 4 provides more details about the structure of the ranger box. The camera is a CCD module producing a video signal. A wide angle lens system with a nominal focal length of 10 mm is mounted on the camera. The parameters of the camera shown in Figure 10 were obtained with the lens system focused at 3 ft for an aperture set at f-11. More complete characteristics of the camera are described in DeMenthon and Asada [11], along with a description of the calibration method which produced the parameters of Figure 10. The video signal is digitized in a VICOM computer, also used to run the software controlling the scanning and range image construction.

The components of the stripe scanning mechanism itself are now described. The 150W light generator itself is not shown in the figures. It is secured away from the robot hand, on a side of the robot forearm. Light is brought from this projector to the ranger box by a 4-ft fiber light guide, which flares to a 2.5" x 0.125" line of fibers. The stripe of light from the optic fiber line is further narrowed to 0.1 mm by a brass slit. The diverging light from this slit is refocused to a planar sheet of light by a 4" long cylindrical lens which has a 44 mm focal length. The light-sheet is redirected out of the ranger box by a 3 in. long mirror, a front-silvered microscope slide. The angular displacement of the mirror is precisely controlled by a geared-down stepper motor. The mirror control system is described in Sections 6 to 10. For numerical values of the characteristic parameters of the ranger, see Figures 5 and 10. For equipment specifications, see Appendix A.

5. Optical performance of the stripe projector

The length of a stripe is 200 mm at a distance of 400 mm from the mirror axis. The thickness of the stripe in the direction normal to the light-sheet is 2 mm at 400 mm. The thinnest stripe one could possibly obtain at a given distance is the size of the image of the slit. For the focal length and slit thickness given here, the thickness of this image would be 1 mm at 400 mm, i.e. one-half what we actually obtained. But this result would be achievable only with an axisymmetric lens, instead of a cylindrical lens. The reason is

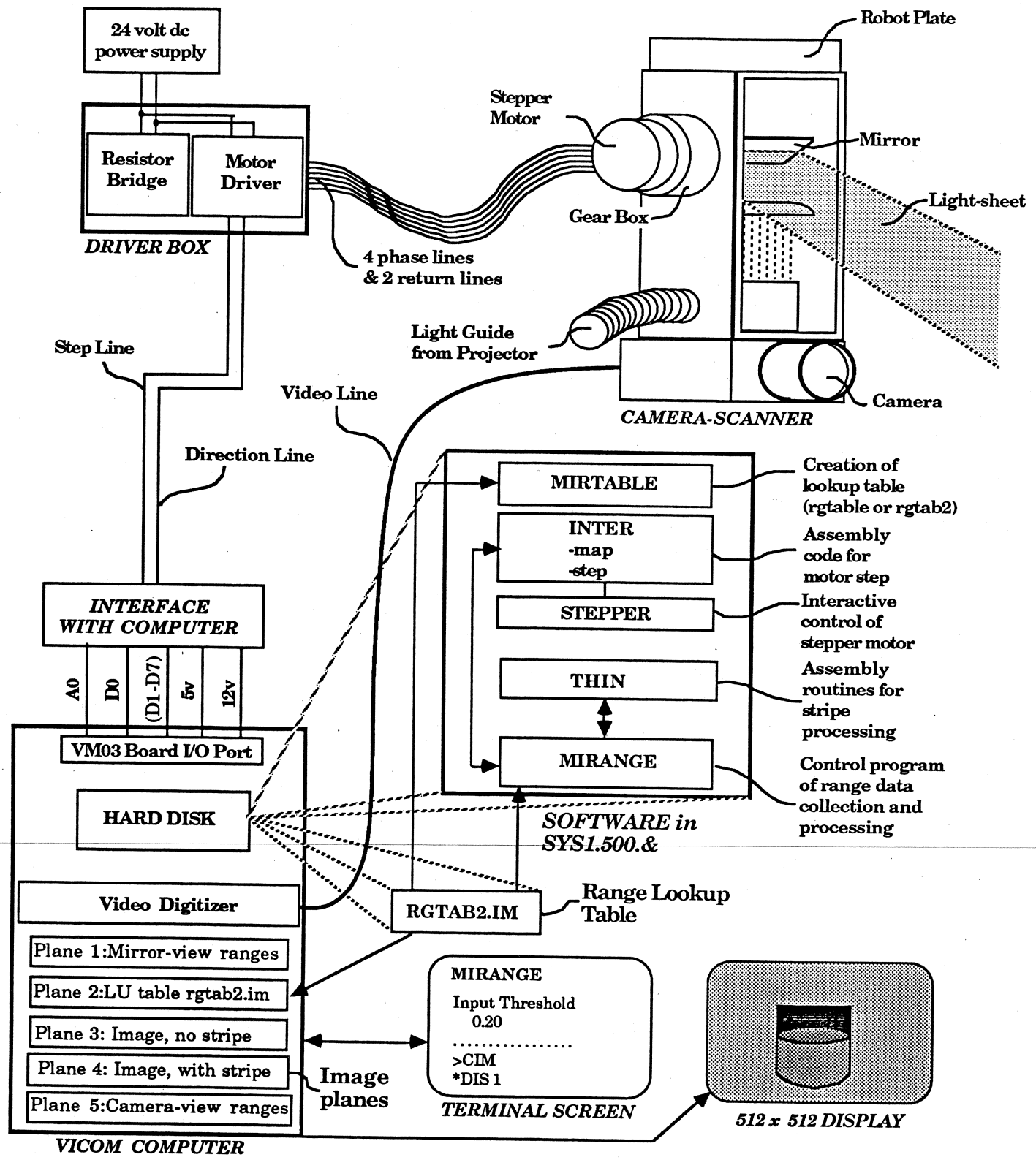

Figure 3: Architecture of the Ranging System (Hardware and Software)

The hardware elements are: Camera-Scanner, Driver Box, Computer Interface, Computer, Terminal and Display Screen. The software modules are: MIRTABLE, INTER (map and step), STEPPER, THIN and MIRANGE.

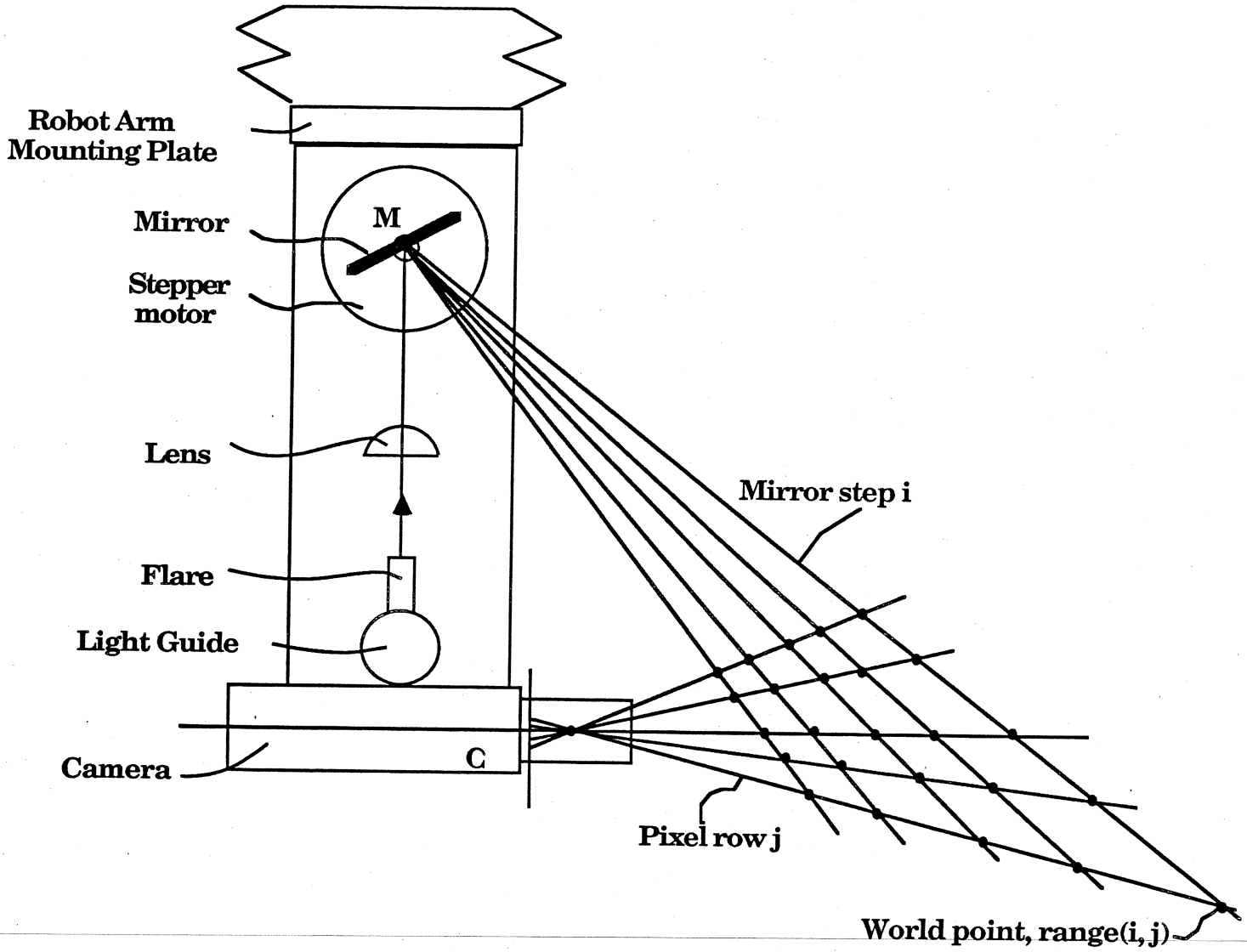

Figure 4 : Structure of Light-Stripe Range Scanner

The angles of planes of light are quantized by the stepper motor. The camera detection of the vertical angular position of a world point illuminated by a plane of light is quantized by the CCD sensor rows. Thus the measurements of the distances of world points to the mirror axis are subjected to quantization. Precise measurements can only be performed on lines at the intersection of the pencil of planes of light and the pencil of planes of sensor rows (defined by the sensor rows and by the camera nodal point). All points of such lines have the same range, if range is defined as distance to the mirror axis. Thus all measurable ranges can be stored in a 2D lookup table indexed by row numbers and mirror steps. This table maps the physical array of intersections of the two pencils of planes, seen as an array of points in this figure.

that in the image of a slit through a cylindrical lens, one cannot have good focusing both for rays taking the shortest route from the slit to the lens and for rays travelling at an angle in the plane defined by the slit and the cylindrical axis of the lens. Unfortunately, an axisymmetric lens would give a 1 mm thick stripe only along the optical axis. Since the slit is very long, the rays from the ends would be subjected to large lens edge distortions.

For lower beam divergence, one would have to resort to a laser source and a cylindrical lens, used this time to spread the narrow beam into a sheet while preserving the thickness of the sheet at the thickness of the original beam. However, the brightness of the stripes would then be much lower, since the power in the beam would be only a few mW, whereas the power in our light-sheet is several W.

One should realize, however, that the occurence of thick stripe images is a generic drawback of light-stripe scanners, regardless of the projection system. Indeed there will always be scene surfaces lit at grazing angles by the light-sheet, producing very thick stripe images.

In any event, in our system only a median curve of the stripe image is used. This median curve is obtained by replacing the two boundary points on the same column of the stripe image by their midpoint. This curve is approximately the image of the part of the stripe created by the median plane of the actual light-sheet. Thus using only the median curve of the stripe is approximately equivalent to working with an ideally thin sheet of light. Errors still occur when stripes have corners, because of the simplification of calculating midpoints column by column. These errors result in a smoothing of the edges of reconstructed objects.

6. Motor and Gearbox

The motor rotating the scanning mirror is a stepper motor rated at 5 percent accuracy. This motor is designed for 1.8 degree steps, but it is operated in half-step mode, and mounted on a gearbox with a gear-down ratio of 1:20, giving angular steps for the mirror of 0.045 degrees, and angular displacements of the light-plane of 0.09 degrees. A 360 degree mirror revolution takes 8000 step commands. These specifications were chosen so that in range images of scenes placed a few feet in front of the ranger, the camera is able to capture the images of at least 512 distinct stripes, giving 512 x 512 range images (Figure 5).

The motor has four windings, of resistance 5 Ω and inductance 9.57 mH each, rated at 1 A. It is powered by a 24 Vdc power supply. The voltage is brought down to nominal values by resistors placed in series with the windings (Figure 6).

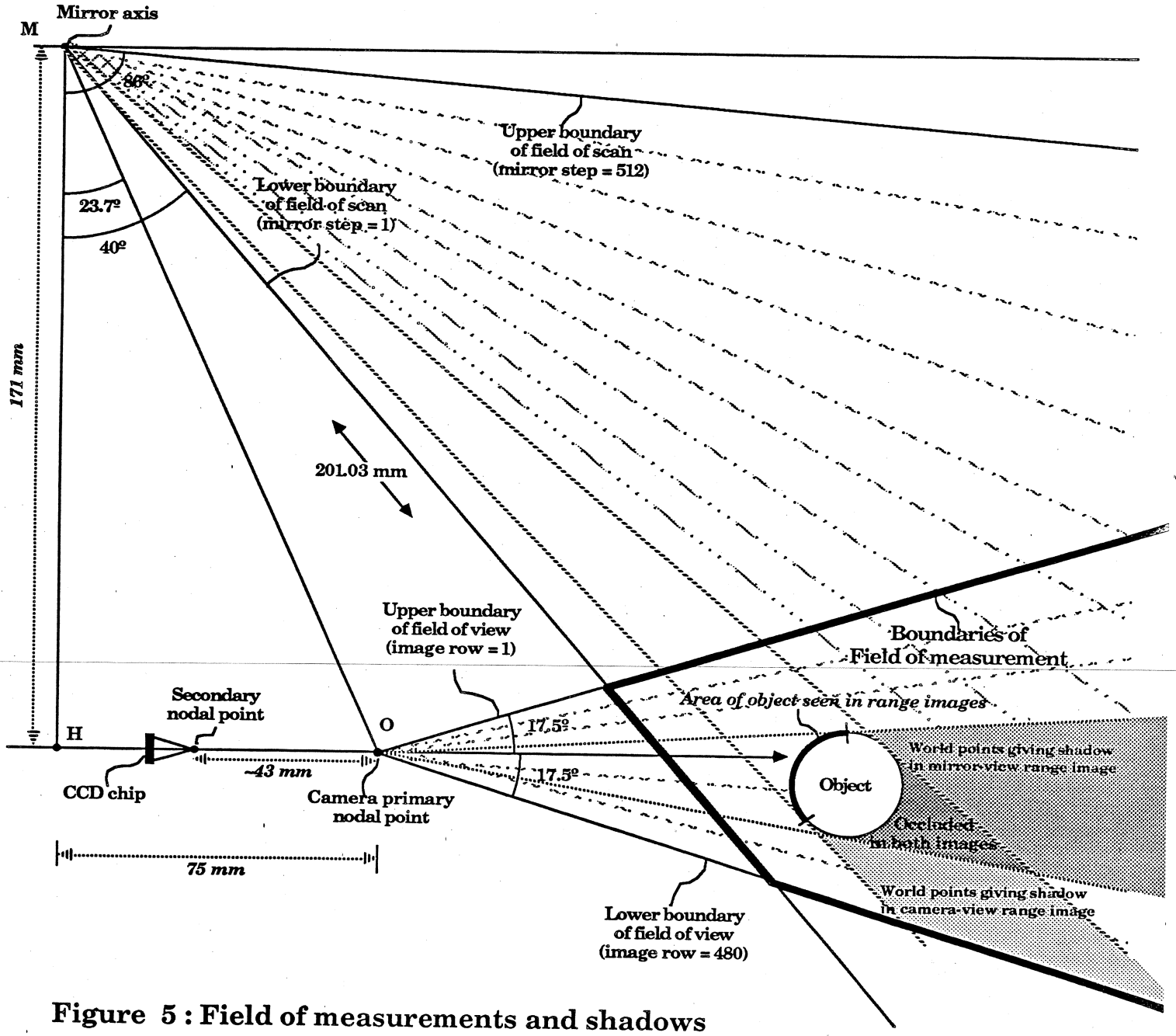

Figure 5 : Field of measurements and shadows

The field of measurement is at the intersection of the camera field of view and the mirror field of scan. Domains of world points giving shadows in the mirror-view range image and in the camera-view range image are shown. Shadows in the mirror-view range image, for example, are defined as images of world points visible from the mirror, but occluded from the camera by the object, and for which the range calculation cannot be performed.

7. Motor Driver

The four windings of the stepper motor, w1, w2, w3, w4, need to be activated in succession for the motor to rotate four successive steps; a sequence w4, w3, w2, w1, and so on, would give the inverse rotation. For half-step rotations, the excitation pattern is w1, then both w2 and w3, then w3 alone, then both w3 and w4, then w4 alone, and so on. A motor driver performs this function. The motor driver (Figure 6) creates this sequence of excitations in the four windings of the stepper motor by sequentially grounding its four output pins 1-4. A new step of the sequence is performed every time a new request for a step is sent on its step-clock pin 11 in the form of a 5 V pulse of at least 15 μs. The sequence order is controlled by the voltage level of the Direction pin 9. Half-step sequences are produced when both pins 7 and 8 are set high. The driver can operate between 15 and 38 Vdc, and is powered here by a 24 V power supply.

8. Interface with computer

The stepping motor is controlled by the VICOM computer (which is also in charge of camera image digitization), by means of the motor driver, and by means of a custom-built interface shown in Figure 7 and described here. This custom interface uses a memory-mapped I/O bus of the VICOM computer. When one writes to specific addresses assigned to a memory-mapped I/O port, the address and data are sent out to this port.

For the control of the stepper motor, the least-significant bit A0 of the address and the least-significant bit D0 of the data are used to convey two elements of information:

- Rotate the motor by one step.
- Reverse the direction of rotation.

Since the interface is asynchronous, Write-Enable, Strobe and Data Acknowledge lines are also used for handshaking. Provision was made in the design to also read 8 bits of data from the outside world back to the computer, which explains why 8 data lines and the Write-Enable lines are hooked to the interface, but this capability was not used here.

The signals required from the custom interface by the motor driver are a 5 V pulse of at least 15 μs to the clock line (11) to trigger a motor step, and a 5 V level to the Direction line (10) for clockwise rotation, 0 V for counterclockwise rotation. We now explain the stages of transformation from the computer signals to the driver input signals.

The IO-channel interface bus of the VM03 processor board of the VICOM was available, and it was used to output the signals to our interface. This bus address and data output maps the computer memory locations between 0x7DE000 and 0x7DEFFF. The signals made available to the interface are

- The Address Strobe (STB*),

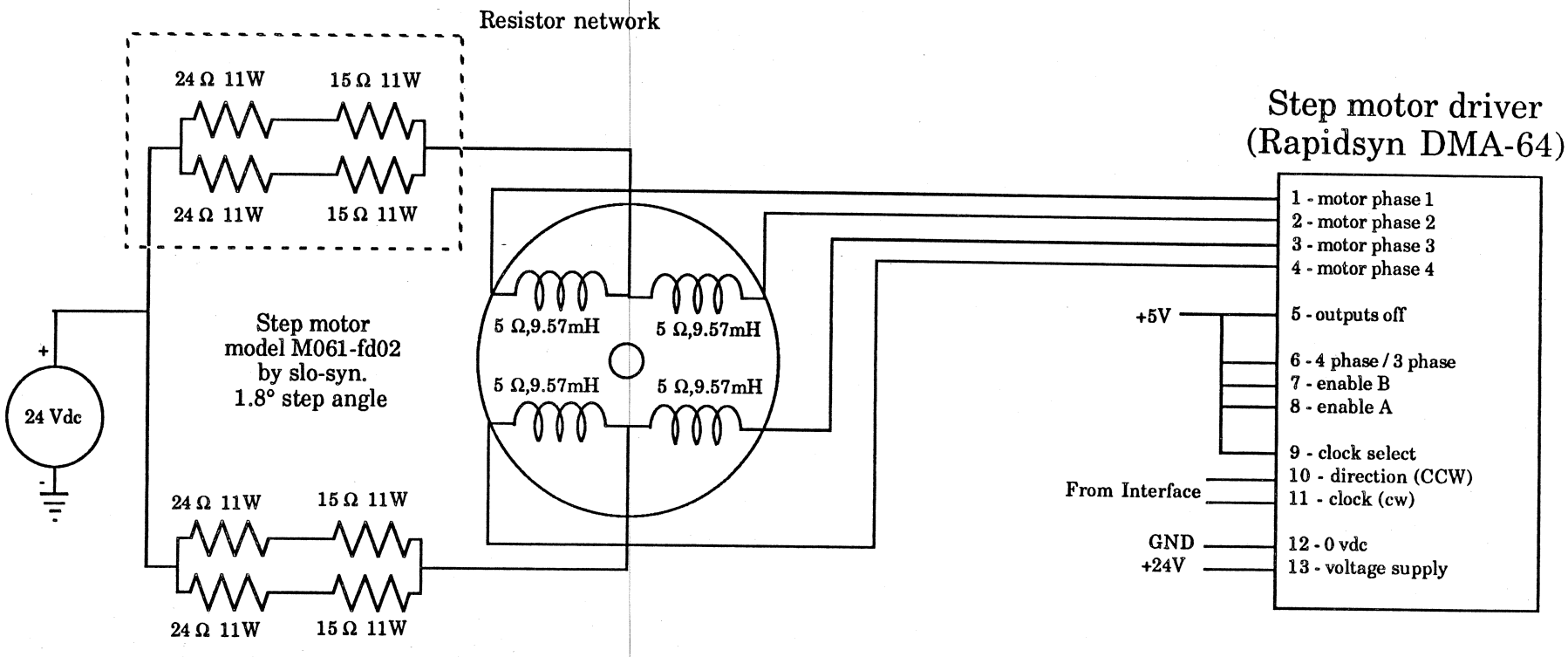

Figure 6: Motor Driver and Stepper Motor

The motor driver receives pulses from the computer interface as commands to rotate the stepper motor. For each step command pulse, the driver activates the next motor stator winding in the sequence, pulling the rotor away from the previous winding.

- Write-Enable (WRI*),
- Data Transfer Acknowledge (XACK*),
- 4 MHz Clock (CLK),
- Data Lines (D0-D7),
- First Address Line (A0),
- 5V and 12V lines.

The other lines of the bus are not used.

There are five sections in the interface:

1. Address Decoder,
2. Data Transfer Acknowledge (Dtack) Timer,
3. Step-Control Clock Generator,
4. Latch Section,
5. Transistor Line Drivers.

These sections are framed by dashed numbered rectangles in Figure 7. The function of each section is now described.

1. The Address Decoder is activated by the Address Strobe Signal, and decodes the least significant bit of the address bus (A0). If A0 is 0, the Step-Control Clock Generator is activated to produce a pulse of around 50 μs duration, which is sent to the motor driver through the Line Driver, and signals "rotate one step" to the motor driver. If A0 is 1, the Latch Section controlling the Direction signal to the motor is enabled.

2. The Dtack timer's purpose is to send an Acknowledge signal back to the computer with a 500 ns delay after the Strobe signal was received. This is done with two JK flip-flops in series, controlled by the 4 MHz clock of the computer, with reset linked to the Strobe.

3. The purpose of the Step-Clock Section is to give a pulse long enough to be acknowledged by the motor driver. The minimum required by the driver is 15 μs. This circuit is composed of a monostable multivibrator with a capacitor and a potentiometer. The potentiometer was adjusted for a 50 μs pulse.

4. The Latch Section consists of two 8-bit transparent latches with tristate outputs. One of the latches is an 8-bit output port and the other is an 8-bit input port. When the Latch Section is selected by a 1 in A0, the Write-Enable line is checked to determine which latch is active. In the present implementation only D0 is transmitted through the output port when a Write operation is in progress, which selects the rotation direction of the motor. The input port is not presently used but could send feedback data to the computer, such as status of switches or position of an encoder.

5. The custom interface is mounted inside the computer casing, while the range scanner is located in a different room. Line drivers were required to transmit the signals along 50 yards of lines to the motor driver, located next to the range scanner.

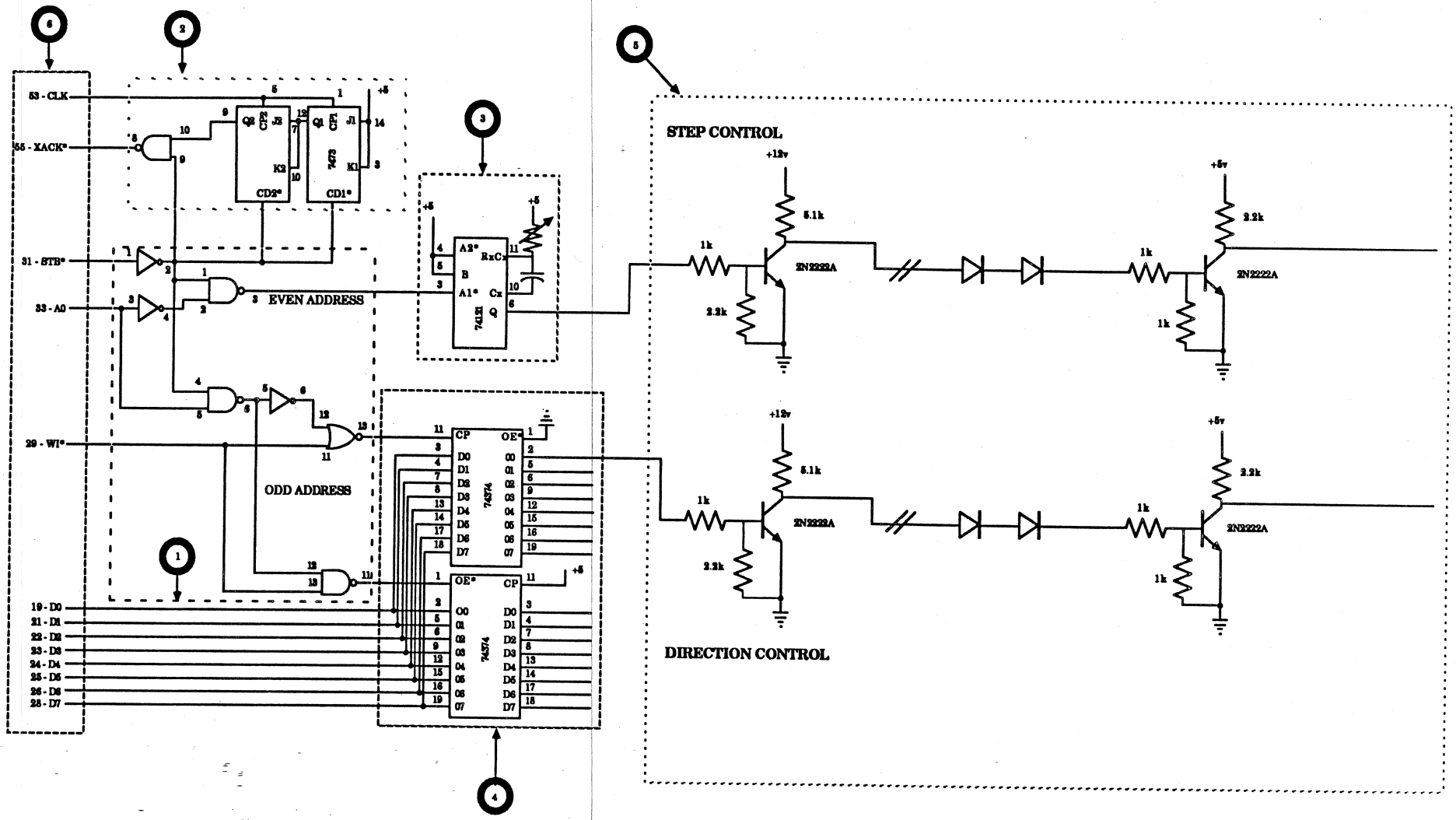

Figure 7: Computer Interface for the Motor Driver

The user can send the command "STEP" from Pascal codes to obtain one rotation step of the motor. This interface is one of the links in the command chain.

The command "STEP" is the name of an assembly code which sends a zero to an even address in a part of the memory dedicated to input/output communications with the port used here. This circuit detects that an even address has been called and sends a pulse with the appropriate characteristics to the Motor Driver Box (see Figure 6).

To reverse the rotation of the motor, odd addresses can be called.

9. Mirror control calls from Pascal programs

The processor of the VICOM computer is a Motorola 68010 with MMU. Function MAP and procedures STEP, PUT, GET are the 68010 assembly language routines which were written to control the range scanner. They have to be declared at the beginning of VICOM Pascal programs, in the following way.

```
function map: integer; FORWARD;
procedure step; FORWARD;
```

A call to function MAP generates the system calls necessary to reserve the memory locations required for the VICOM I/O (see Appendix B). A call to procedure STEP rotates the mirror by one step. The assembly language in STEP calls an even address within the block of the I/O and puts a zero at this location:

```
XDEF STEP
EVENADDRESS EQU $7DE000
STEP   MOVE.B  #00, EVENADDRESS
```

Procedure PUT can be used to change the direction of the mirror rotation, and GET reads inputs from the interface. (**Note**: These two calls are not fully implemented. PUT is not used because it is preferable to always rotate the mirror in the same direction to avoid gear lash. GET is not used because the mirror positioning obtained by the stepper motor is reliable and repeatable, and does not requires any feedback from the ranger to the computer.)

10. Home Position of the mirror

The starting position of the mirror (*home position*) is taken to be parallel to the optical axis of the camera, so that the initial position of the plane of light is normal to the optical axis.

This position was chosen because it is checked easily with the light projector turned on. In home position, the mirror sends the light back to the slit, and the stripe visible on the brass slit plate is aligned at the top of the slit. The mirror comes back automatically to its home position at the end of the range image production (program MIRANGE, see Section 17) and should be at its home position before starting acquisition of a new image. If the mirror has to be repositioned, the program STEPPER allows interactive mirror control. See Section 20 for details.

Obviously, the calculation of ranges for triangulation cannot start with the mirror at home position. Triangulation can be done only as soon as the sheet of light cuts the camera field of view. When the user starts the program MIRANGE, the mirror is rotated to an *initial operating position*, 20 degrees away from the home position, giving a rotation of 40 degrees for the light-plane. Figure 5 shows a side view of the space of possible range measurements, at the intersection of the camera field of view and the mirror field of scan.

11. Overview of the range computation method

A light-sheet is created by the stripe of optic fibers, focused by a cylindrical lens, and rotated to an angle ϕ from its home position by a computer-controlled mirror. It intersects an object in a planar world stripe, and the image of this stripe is seen from the camera and is usually curved (Figure 8). This image stripe actually has a thickness of several pixels, and the median curve of this stripe is obtained; for each pixel of this median curve the row in the image is detected. The world point corresponding to this pixel belongs to an image row plane defined by the pixel image row and by the camera image center, and the concurrent knowledge of the angle of this image row plane and the angle of the light-plane allows calculation of the specific distance from the world point to the *mirror axis* by triangulation. In practice these calculations are performed only once and for all for the ranges to the mirror axis of all the intersections between the pencil of image row planes and the pencil of light-planes within the camera field of view and the mirror field of scan (Figure 4). The results of these calculations are placed in a two–dimensional lookup table. The next section gives more details on the construction of this table and the calculation of the ranges.

12. Justification for axial ranges and a 2D range lookup table

As we just mentioned, the range lookup table lists ranges from the mirror axis instead of ranges from the center of the mirror. The reason is that all the measurable axial ranges can be stored in a 2D table, whereas a 3D table would be required if ranges from a point were used. Indeed all the world points which have their images on the same image row when illuminated by the same plane of light have the same axial range, independently of their image columns. Thus the table needs only be indexed by the mirror steps and the image rows, not the camera image columns. To see this, consider that world points which have their images on the same row when they are illuminated by the same light-plane are at the intersection of the light-plane with the image row plane defined by the image row and the lens center (Figure 9). This image row plane is parallel to the mirror axis because the mirror axis was made parallel to the image rows; thus the intersection between this plane and the light-plane is also parallel to this axis. Therefore the world points of this intersection have the same axial range. Since there are only limited numbers of image rows and mirror steps, all the possible axial ranges can be precalculated and stored as elements of a 2D array indexed by the mirror steps and image rows.

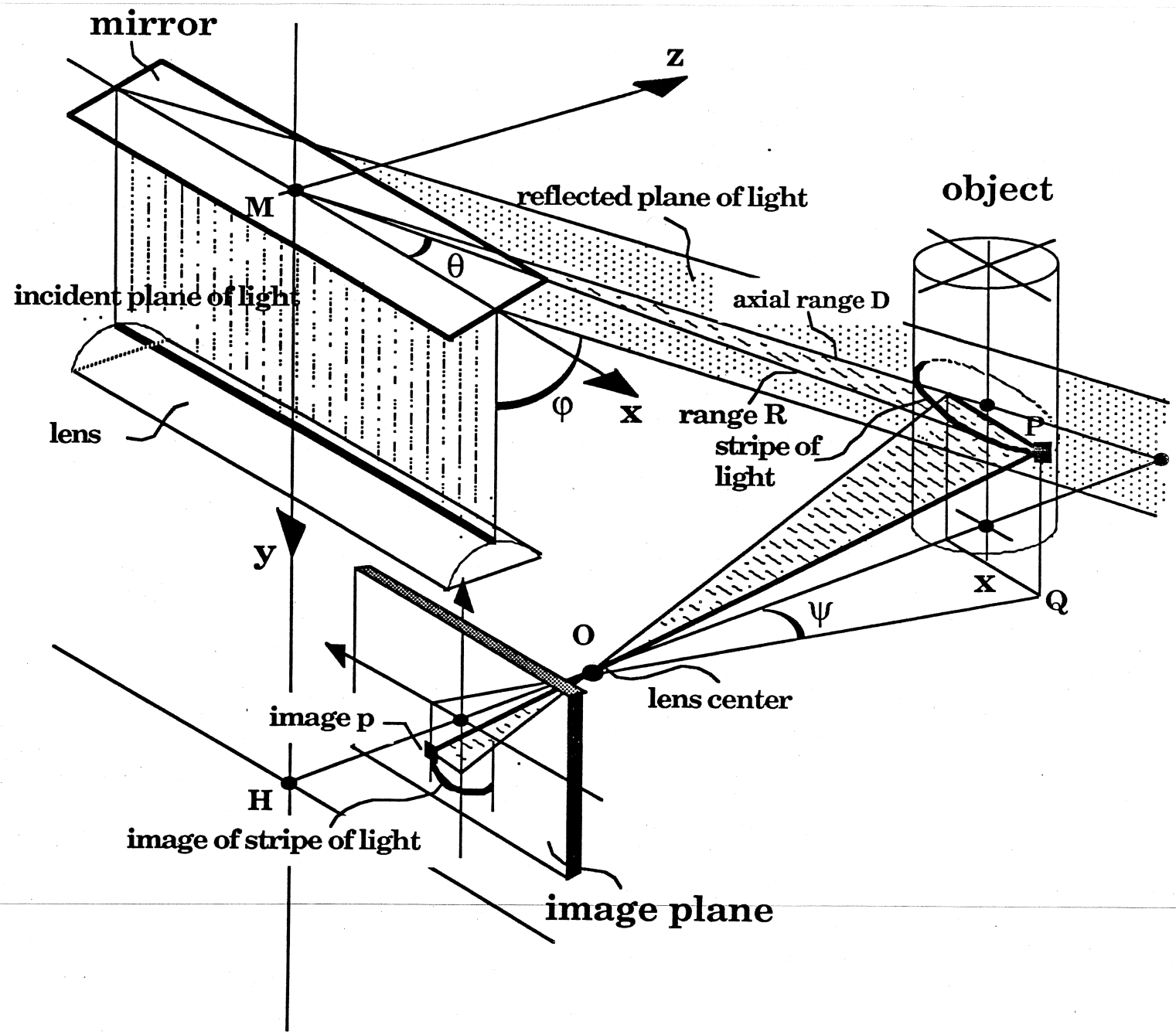

Figure 8: Perspective view of geometric relations between world scene, mirror angle, plane of light and image of stripe on image plane.

In a mirror-view range image, the column numbers of the pixels are the original columns of the images p of world points P, while the row numbers are the mirror step numbers, related to ϕ, and the grey levels represent the range D. From the column number, one can deduce ψ, and with ϕ and HO, calculate the Cartesian coordinates of P.

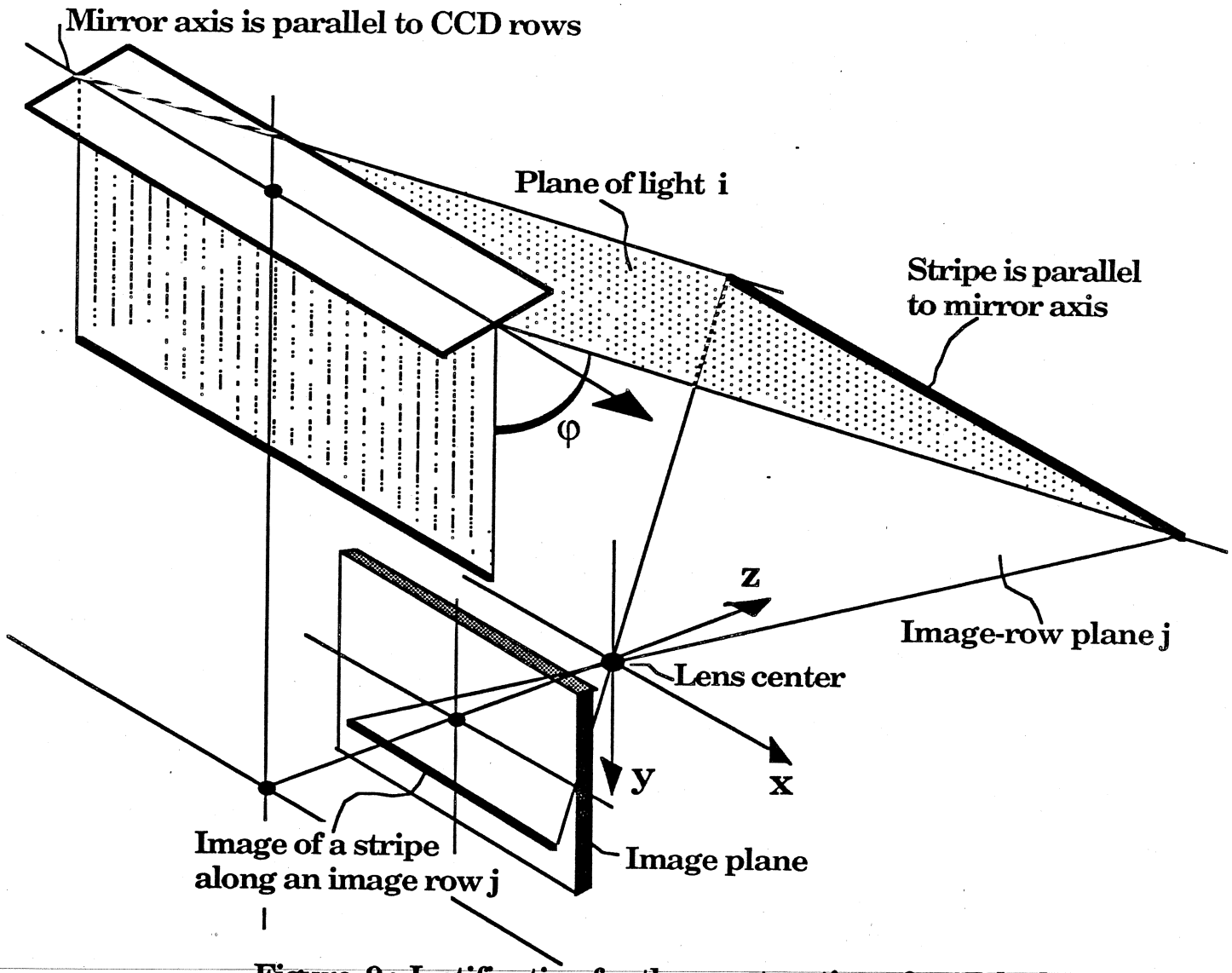

Figure 9 : Justification for the construction of a 2D lookup table for ranges defined as distances to the mirror axis.

For a given mirror step i, image points belonging to the same image row j correspond to world points which have the same range with respect to the mirror axis. Indeed, such world points belong to a line parallel to the mirror axis.
This is due to the fact that the mirror axis is set parallel to the camera sensor rows. Thus a plane containing a row is parallel to the mirror axis, and the world points are at the intersection of such a plane and the plane of light.
Therefore one just needs to build a 2D range table indexed by the mirror steps i and the image rows j. The ranges in this table are distances to the mirror axis and not distances to a point.

13. Calculation of axial ranges

Now details are given for the calculation of the axial range D of a world point P on a stripe created when the plane of light is generated at step i of the mirror. This is the calculation performed to fill up the range lookup table. It assumes that the specific row of the image of the point P has been given, by an image processing routine described in the next section. Figure 10 shows the relative positions of P, the mirror and the camera, in an orthogonal projection in the direction of the mirror axis. In such a projection, distances to the mirror axis are conserved; thus the calculation of range with respect to the mirror axis can be done in the plane of the figure. O is the lens center, M the projected location of the mirror axis. The range calculation applies the classic *law of sines* in the triangle MOP. The length OM is a parameter defined by the structure of the system, and the angles α and β are defined respectively from the row number of the image of P and from the mirror angle.

$$D = \frac{OM \sin \alpha}{\sin(\alpha + \beta)}$$

α is expressed in terms of known parameters:

$$\alpha = \frac{\pi}{2} + \gamma + \delta$$

with

$$\delta = \text{Arctan}\left[\frac{row - C_y}{f_y}\right]$$

in which

row is the row number of the image of P in screen pixels,

f_y is the focal length in the y-direction in screen pixels,

C_y is the row number of the image center in screen pixels,

β is now expressed in term of known parameters:

$$\beta = 2\,m - \gamma$$

where γ is the angle which the line OM makes with the starting position of the plane of light and m is the angle given to the mirror from it home position to the position which gave the plane of light illuminating the world point P.

$$m = stepangle\,(n_o + i_1 - 1)$$

Here, *stepangle* is the angle of one mirror step. The total number of mirror steps from

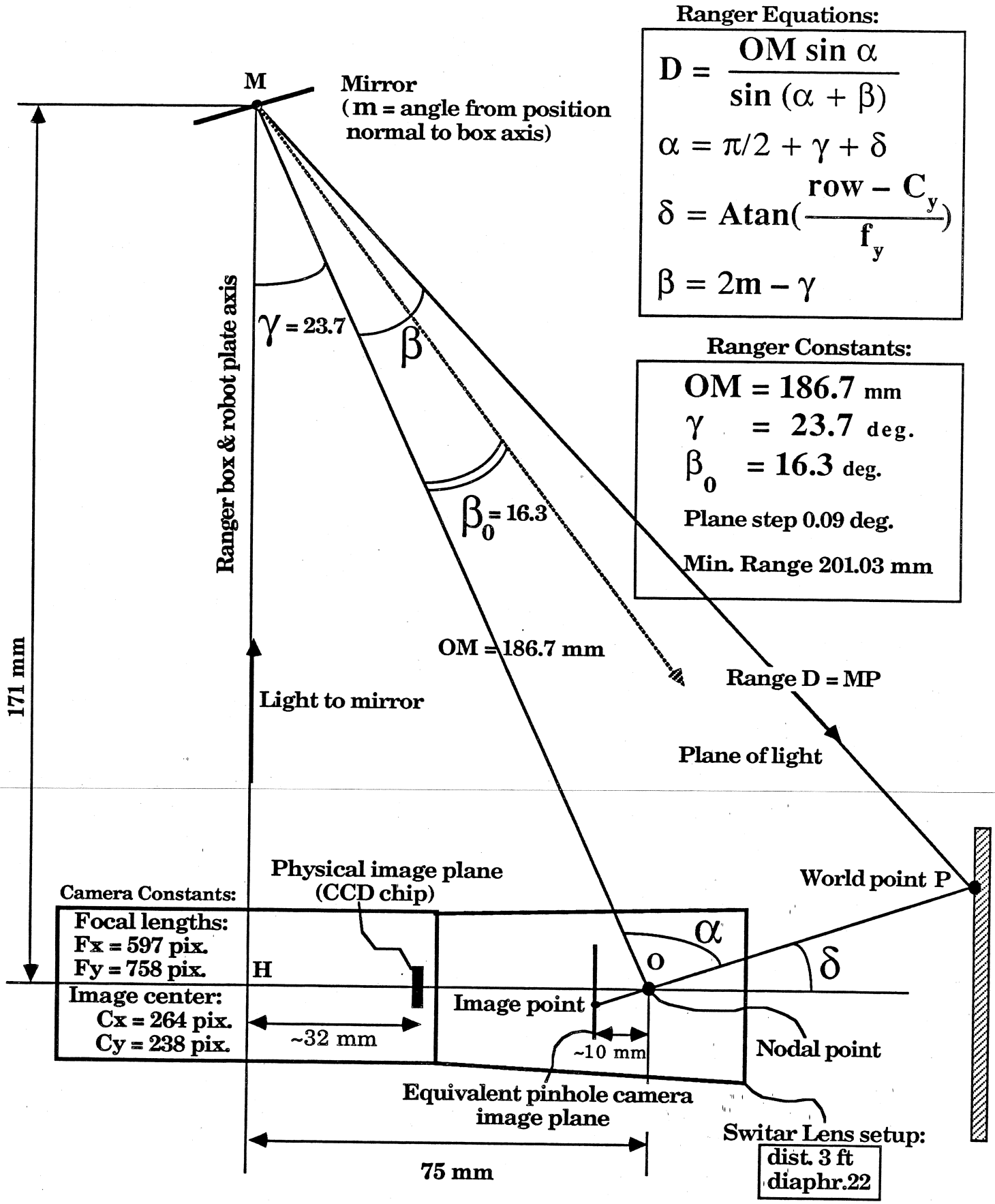

Figure 10 : Ranger Calibration Parameters and Variables

home, i, has been decomposed into two numbers, n_o and i_1. n_o is the number of steps which bring the mirror to its initial operating position (Section 10), and i_1 is the number of mirror steps from this initial operating position.

In our design, *stepangle* = 0.045 degrees. The initial operating position is for a mirror angle of 20 degrees and a light-plane angle of 40 degrees from home position. This corresponds to an angle β (denoted β_o in Figure 10) of 16.3 degrees for the light-plane. The number of steps n_o is 444. The values of OM, γ, C_y and f_y obtained by the calibration procedure described in DeMenthon and Asada [11] are listed in Figure 10.

14. Format of the lookup table

The range lookup table is a 512 x 512 array. There is one row per mirror step angle; the first row corresponds to $i_1 = 1$. There is one column per image row. The first row corresponds to the first image row. The fillup process takes place in the VICOM Pascal program called MIRTABLE which functions as follows:

```
Begin
For pixel row r from 1 to 512
    Calculate angle a_r {from Section 13}
    For mirror step i_1 from 1 to 512
        Calculate angle b_i1
        Calculate axial range D = f(a_r, b_i1 )
        Code the range D to a grey level and put value in element (i_1, r) {Next Section}
    End for
 End for
End.
```

Source code for MIRTABLE is given in Appendix B.
We now examine the method of coding the ranges.

15. Range coding

The range calculation of Section 13 outputs distances from world points to the mirror axis in mm. It is useful to code these ranges in a format which makes them suitable for display on a graphics system, such as the GRINNELL or VICOM systems of our lab. Indeed the lookup table data are used to build range images, and one must be able to display these range images in the same fashion as regular brightness images. Therefore

the coding of the range must serve two purposes:

• Allow the display of all pixels whose values represent ranges within the range domain of interest,

• Show good contrast between pixels representing different ranges to permit easy visual differentiation between ranges on the display.

Furthermore, if larger ranges are represented with darker grey levels, the coding function must be a *decreasing* function of the range. To satisfy the previous conditions, the closest ranges must correspond to all the bits turned on in the grey level data, and the furthest ranges must turn off all the data bits. The pixel grey level is represented in the upper 8 bits of the 16 bit pixel data for the GRINNELL displays and 11 of the upper 12 bits in the 16 bit pixel data for the VICOM display.

Coding by a decreasing *linear* function is attractive because the decoding from range image to exact range is fast, a critical factor if the decoding is done on all 262,144 elements of the 512 x 512 range array. The only drawback is that the function has to be set to zero beyond a certain range, or it would become negative, and would give unexpected results with the unsigned integers of the pixel grey levels. For the GRINNELL display, the chosen linear coding function is:

```
maxrange := minrange + 255 {maximum codable range}
if range>0 and range <= maxrange then
    code := 256 * trunc(maxrange – range)
else
    code := 0;
```

The multiplicative coefficient 256 (2^8) shifts the result to the 8 upper bits of the pixel data, which are the bits dedicated to the pixel grey level among the 16 pixel bits.

The minimum measurable range, *minrange*, is set equal to the ranges of world points illuminated by the first scanning position of the plane of light and having their images on the first row of the CRT. For our configuration, this minimum range is 201.03 mm (Figure 5). The maximum range, *maxrange*, is set at 255 mm away from minrange.

The grey byte is set equal to trunc(*maxrange – range*). This expression is equal to 255 when the range is minimum, corresponding to all 8 bits set to 1, and decreases linearly as the ranges increase. Beyond a range equal to *maxrange* all bits are zero; therefore world points with a range larger than 255 mm beyond *minrange* are coded with a zero value (black on the range image).

To prevent this drawback of linear coding, one can use a coding function which tends asymptotically towards zero for large ranges, while having a large slope in the ranges of interest. The decreasing S-curve given by a Gaussian type of expression satisfies these requirements. The range around which we want the maximum slope is called midrange:

```
code:= round (16 * 2047 * exp (- 0.5 * sqr((range – minrange) / (midrange – minrange)))
```

The exponential term itself decreases from 1 for a range equal to *minrange* to zero for infinite range, with an inflection point at *midrange*. The coefficients in the expression apply to the VICOM data format. The multiplication by 2047 is used to fill up 11 of the 12 grey level bits when the exponential is one, and leave the sign bit alone. The multiplication by 16 positions these bits as upper bits in the 16 bits of pixel data. The only drawback of this method is the longer decoding time from grey levels to ranges in mm than with a linear coding.

The program MIRTABLE creates the lookup tables for either of the two range codings described above. After starting this program, the user is asked to make a choice: **"Type 1 for a nonlinear range coding, 2 for a linear range coding"**. If the first option is chosen, the lookup table "RGTABLE.IM" with S-curve range coding is created. If the second option is chosen, the lookup table "RGTAB2.IM" with linear range coding is made. The program MIRANGE, which controls range data collection with the scanner, makes exactly the same inquiry when it is started. It then uses the lookup table corresponding to the answer. Of course, MIRTABLE should be run only to recreate new lookup tables if MIRANGE does not find a lookup table on hard disk, in cases where data have been destroyed.

16. Range image formats

The control program of the range scanner produces two range images, a *camera-view* range image and a *mirror-view* range image. The distinction between these two methods for building range images from light-stripe range finders has been summarized in Section 2, and is detailed here.

Assume that a world point P has been illuminated by a plane of light created at mirror step i, and is visible at a pixel p of the camera image at column j and row r. The knowledge of i and j allows for the calculation of the range from P to the mirror axis (Section 13), or the reading of the range in a precalculated lookup table (Section 14). This range is then coded to match the grey level formats required for regular images (Section 15).

Camera-view range image:

To create a camera-view range image, this coded range is placed at the position occupied by the pixel p in the original camera image. This range image format is the classical format used in most previous work, e.g. Agin [5], Shirai[6]. Since a pixel p belongs to the image of the stripe, the resulting image array is an image of all the stripes with grey levels coding the ranges, and zeros (no data) between the stripes. It is therefore a very sparse array. It would be difficult to do any processing directly on this type of range image, such as smoothing and edge detection. Also, 3D reconstructions from these images are slightly inconvenient, because they require going over all the useless pixels to pick the useful pixels and calculate 3D coordinates from their range code.

Mirror-view range image:

To create a mirror view range image, the coded range is placed in an image array at the same column j as the pixel p, but at a row i_1 equal to the mirror step number, counted from the bottom of the array. Indeed the direction of the mirror rotation is such that the first mirror steps scan the base of objects, and the range of these points is put at the bottom of the range array. The rows corresponding to the lower mirror step counts are at the bottom of the array, so that the base of an object is found at the bottom of the range image.

The result is a densely populated array, on which smoothing and edge detection can be performed with standard techniques. Pixels p from the same stripe are found on the same row. If we placed a camera with its lens center on the mirror axis, all the stripes would be seen as straight lines in the image plane of this camera. It would be easy to thin each stripe, and shift them to eliminate the gaps between them. Then the brightness levels would be replaced by ranges of the corresponding range code. The result would be a range image very similar to the image that we synthesize from the actual camera. This justifies the name "mirror-view range image" given to this range image. Notice, however, that the column position of a pixel in our range image is still the column position of the pixel in the actual camera image, not the column position in a virtual camera at the mirror position. This hybrid format simplifies the filling in of the range array and does not complicate the calculations in 3D reconstructions.

This second range image format has the other advantage, already mentioned in Section 2 and Figure 2, of being closely related to the range images obtained by a laser range scanner. In a laser scanner range image, pixels in the same row also correspond to world point which are scanned when the mirror horizontal axis is at a given step. With the laser ranger, however, the columns of the array are indexed by the steps of the other mirror doing the scanning, i.e. any two adjacent pixels of the same row correspond to two laser beams making an angle equal to two times that mirror step. Therefore there is a horizontal distortion between the laser range image and the light-stripe range image, leading to different reconstruction formulas for the 3D reconstruction.

17. Range image acquisition

The ranger is controlled by the Pascal program MIRANGE.
The organization of this control program follows.

Begin

Ask whether nonlinear or linear coding of range is desired.
Load the corresponding lookup table from disk to image plane 1.
Ask for input of a threshold level between zero and one.

{Proper thresholding should keep the whole stripe of light in the camera image, and nothing else. A value of 0.20 when the diaphragm of the camera is completely closed seems to give good results most of the time}

Digitize camera image and send to image plane 3 {image without stripe}

{Go to the position from which range scanning will be started, requiring n_o steps:}

For i := 1 to n_o

STEP {one motor step, see Section 9}

Delay {delay loop to accommodate response time of driver and motor}

End for

Digitize camera image and send to memory plane 4 {image with stripe}

{To make sure that mirror is done rotating when digitizing, process previous digitized image while mirror is getting positioned for next image}

{Scan for ranges:}

For i := 1 to 512

STEP {routine which rotates motor by one step}

{Process image obtained after previous digitizing:}

THIN **{Details in next section}**

{THIN puts mirror-view range image pixels in image plane 1 and camera-view range image pixels in image plane 5}

Digitize camera image and send to memory plane 4 {with stripe}

End for

{Complete 360 degree rotation to return to home position:}

For i := 1 to $(8000 - n_o - 512)$

STEP

End for

End.

The program THIN detailed in next section dispatches the range pixels to the mirror-view and camera-view range images.

The mirror-view range image can be displayed by sending image plane 1 to the display monitor.The camera-view range image is found in image plane 5. Directions are given in Section 20.

Source code for MIRANGE is given in Appendix B.

18. THIN routine

Every time the mirror is stepped and creates a new stripe on the world scene, the camera image which contains the image of the new stripe is grabbed and its address is sent to the routine called THIN. This routine scans the image column by column, from the position of the previous stripe and going up in the image, to locate in each column the

lower and upper bounds of the stripe (Figure 11). A lower bound of the stripe is detected when the difference between the no-stripe image pixel and the stripe image pixel is above a specified threshold. At the upper bound of the stripe this test fails again. The pixel row of the midpoint between these bounds is determined, and its range is read in the range lookup table. The lookup table is indexed by the pixel position in the image column and by the mirror step count. Finally, the pixel with its grey level coded by range (Section 15) is added to the two range images described in Section 16. In the first range image (camera-view range image), the pixel is put back at its original position in the camera image. In the second range image ("range image from mirror"), the pixel is placed in the same column where it was found in the camera image, but at a row number equal to the mirror step count, starting from the bottom row.

This algorithm assumes that increasing mirror steps move images of the stripe up in the camera image. In other words, the bottom edge of the current stripe is assumed to be above the bottom edge of the stripe processed at the previous step. The mirror is rotated with respect to the scene in such a direction that this is verified in most of the cases. This hypothesis greatly reduces the time of search of stripes in the picture. However, in the cases of objects flying above the ground, it could happen that a sheet of light *above* the object would make a stripe on the ground which is visible *under* the object, in the gap between the ground and the object. Thus this strategy would give larger unexplored areas (shadows) on the ground than a more complete search for stripes.

To limit the number of false detections of stripe pixels, a thickness of at least two pixels is required for a stripe. Similarly, a drop in brightness is considered a top edge of the stripe if this drop is maintained for at least two succeeding pixels in the scanned column.

The THIN procedure is an assembly routine called from the Pascal control program MIRTABLE. It is given in Appendix B along with a more readable pseudo-code following the same organization.

The call to THIN follows the syntax

thin (*range, image, stripe, imir, thresh, result, result2*)

where

range is the address of the first element of the range lookup table in memory,
image is the address of the digitized camera image without any stripe,
stripe is the address of the digitized camera image with the stripe to be processed,
imir is the mirror step count,
thresh is the threshold value between 0 and 1,
result is the address of the range image as seen from the camera,
result2 is the address of the range image from the mirror (one row per stripe).

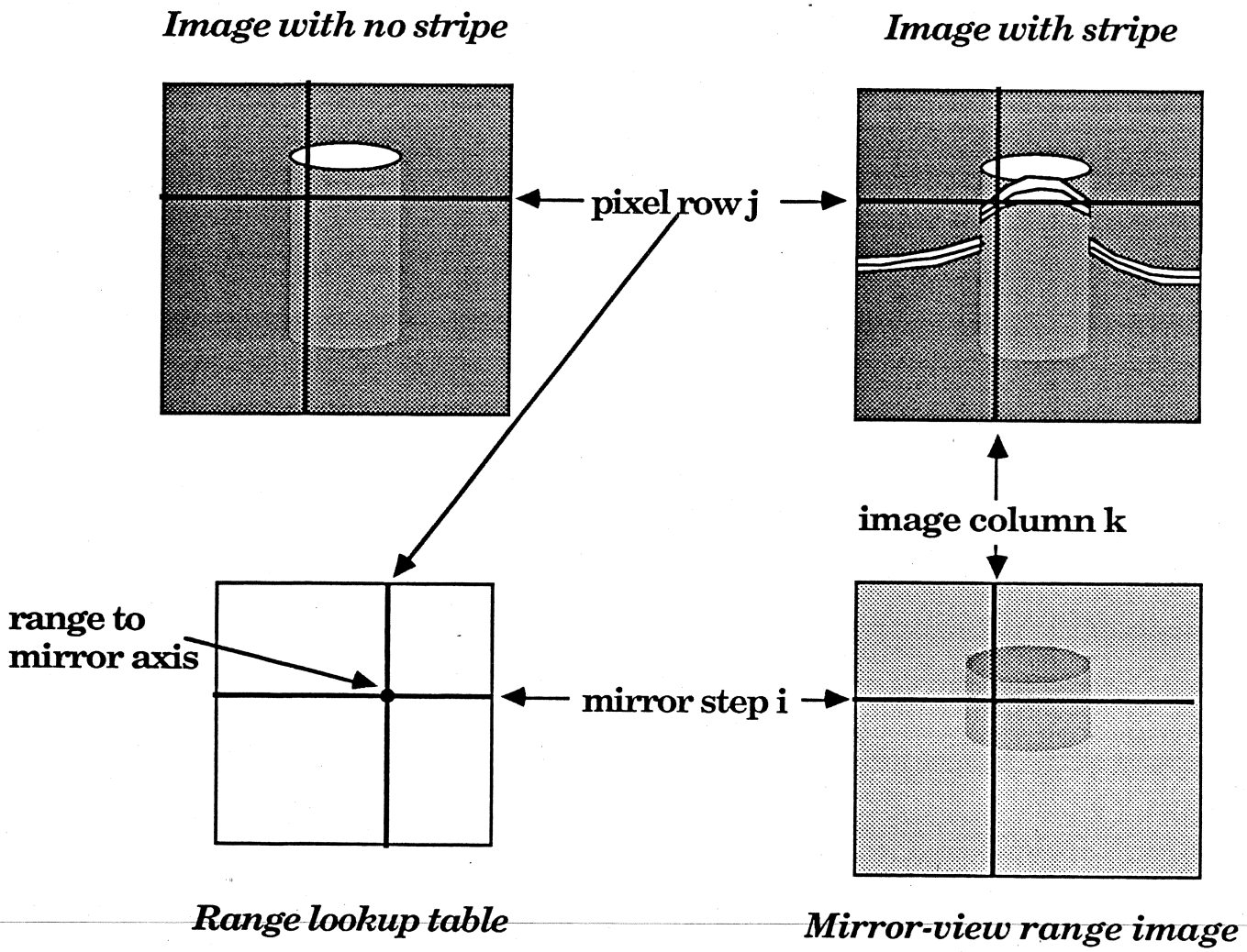

Figure 11 : Creation of a mirror-view range image from images of light stripes

The image with stripe and the image without stripe are simultaneously scanned upward column by column. A pixel at the bottom of the stripe is detected when the difference between the grey levels of the two images is above a given threshold. Then the scan is extended until the difference falls back under the threshold, which gives the top pixel of the stripe. The row of the midpoint is deduced, and used along with the mirror step count to read in a lookup table the range of the corresponding world point with respect to the mirror axis. This range is coded and placed in the mirror view range image at the same column position as the original image pixel and at a row equal to the mirror step count.

19. From ranges to Cartesian coordinates

This section explains how to use a range image to build a Cartesian representation of the world. The goal is to find the Cartesian coordinates for all the world points P which have a projection p on the range image. Only the transformation of the mirror-view range image is considered here.

Cartesian coordinates are expressed in a mirror-centered coordinate system, which is shown in Figure 8:

- The origin M of the coordinate system is the projection of the camera lens center on the mirror axis. It is also the intersection between the mirror axis and the plane of symmetry of the scanner box.
- The x-axis is the mirror axis.
- The z-axis is taken parallel to the camera optical axis.
- The y-axis completes the orthonormal axis system.
- The angle ϕ is the angle of the plane of light with respect to the y-axis. The angle θ is the angle of the line MP from the origin M to a world point P with the x-axis (mirror axis). These angles, together with the distance MP, are spherical coordinates of M. They are also used in reconstruction of scenes from laser range images. The angle θ projects on the yOz plane into an angle φ which is calculated from the column position of the image p of P.

In the mirror-view range image, rows are mirror step numbers, with the last row corresponding to the first scanning step; column numbers are camera column numbers (Section 16). The steps of the transformation are presented, using when possible the same notations as in the range calculations (Section 13):

1. Go back from grey level of the range image pixel p to *axial* range D of the world point P, using the relevant decoding formulas which invert the codings of Section 15:

For linear coding:

If *code* = 0 then D unknown (shadow, or pixel out of field of scan, or range too large)

else D:= *minrange* + 255 – *code* / 256 (with *minrange* = 201.03 mm)

For nonlinear coding:

If *code* = 0 then D unknown (shadow or pixel out of field of scan)

else D := *minrange* + (*midrange* – *minrange*) * sqrt (2 * ln (16 * 2047 / *code*))

2. Go back from row index *row* of image of P in range image to angle ϕ of mirror:

Mirror step number:

$$i_1 = 512 - row + 1$$

Mirror angle:

$$m = stepangle\ (n_0 + i_1 - 1)$$

Plane of light angle:

$$\phi = 2\,m$$

In our implementation, ϕ can be written:

$$\phi = 40 + 0.09\,(512 - row + 1)$$

3. Using ϕ, go back from column index *col* of image of P in range image to coordinate x of P in mirror coordinate system:

Let HO be the z-coordinate of the lens center in the mirror-centered coordinate system (Figure 8). (The absolute value of HO is also the distance of the lens center to the home position of the plane of light).

The coordinates of the image center are C_x and C_y, and the image of P is at a column number denoted by *col*. Q is the projection of the world point P to the plane parallel to xOz going through the lens center, and ψ is the angle of OQ with the optical axis (Figure 8):

$$\tan\psi = f_x / (col - C_x)$$

and the x-coordinate of P can be written

$$x = (D \sin\phi - HO) / \tan\psi$$

or

$$x = (D \sin\phi - HO)\ (col - C_x) / f_x$$

4. The other two coordinates are simpler to obtain:

$$y = D \cos\phi$$

$$z = D \sin\phi$$

In our implementation, the calibration of the system gave (see Figure 10 or [11])

$HO = 75$ mm,

C_x = column of camera image center = 264 pixels,

f_x focal length of camera in x direction, in pixels is equal to 597 pixels. f_y is not used here.

If instead of the Cartesian coordinates, we need the spherical coordinates of P, we have already found ϕ, and we need to calculate the angle θ. The tangent of θ is deduced from the coordinate x_P of P and from its axial range D (Figure 8):

$$\tan\theta = D / x_P$$

We also need the range R from the point M (we have used only the range D from the mirror axis so far):

$$R = D \sin\theta$$

20. Instructions for the use of the light-stripe range scanner

1. Look at Figure 5 to determine the field in which range data acquisition is feasible. See also DeMenthon and Asada [11] for the values of systematic and random errors in the field of measurement.
2. Position the robot arm accordingly with respect to the scene to be scanned.
3. Set the camera lens to 3 ft and f-11 (settings for which it was calibrated).
4. Remove the plate protecting the mirror. Plug in the light guide end to the light projector and tighten the set screw. Turn on the projector and the power supply of the motor interface (feel its vibration).
5. Type: USE SYS1:500.& on VICOM terminal.
6. Check that the mirror faces the slit, and that a stripe is created on top of the slit of the brass plate screwed to the light guide flare (*home position*). If the mirror is in any other position, you have to bring it to this position. To do that, type "STEPPER" and hit return. Then to answer the request "Enter nb of steps", experiment with large numbers, then smaller as you get closer, remembering that the mirror rotates in only one direction and takes 8000 steps for a full 360 rotation.
7. Type: MIRANGE
8. For guidelines on answering the first request "Type 1 for a nonlinear range coding, 2 for a linear range coding", refer to Section 15.
9. To the next request "input threshold, between 0 and 1", answer 0.2. Experiment between 0.1 and 0.3 if the range images are not satisfactory. When looking for stripes in an image, the program calculates the difference between the grey level of a pixel in the image with the stripe and the grey level of a pixel in the same image but without a stripe, and considers that the pixel is in the stripe if this difference is above the threshold. Thus a low threshold will keep more stripe pixels but possibly more noise. To compare results, the camera-view range image, with all the stripes distinctly visible, may be more useful than the mirror-view range image. See step 11 for display of these images.
10. After answering the previous request and hitting return, the scanning of the light-sheet through the scene starts. The numbers printed on the screen are the numbers of steps left from the present mirror position to the home position. The program can be left to run to a full scanning of the scene. Then it will complete its rotation to come back to its home position, facing the slit. In order to shorten the scanning time, if enough of the scene has been scanned, hit the "Break" key to terminate the MIRANGE program. Then use the program STEPPER (see step 6) to send the mirror back to home position, typing in the number of mirror steps indicated by the last number printed by MIRANGE.
11. To display the range images: Type CIM to put the VICOM in interactive mode. Then type DIS 1 to see the mirror-view range image (image plane 1 is displayed), and DIS 5 to see the camera-view range image (image plane 5 is displayed). To see the range-coded grey levels in pseudo-color, type PSE(1) or PSE(2). For display of all the significant pixels of the linearly coded images (option 2 on step 8) shift the bits to the right: Type TWO 1>1 to shift image plane 1 and TWO 5>5 to shift image plane 5.
12. To transfer these range images to the VAX, use the *vreadimge* command from the VAX.

21. Applications of the CVL light-stripe range scanner

Laser range scanners are bound to become popular sensors for robotic systems, as their price and size decreases. They can produce at a high rate images for which the reconstruction of the world points from image pixels is evident, without any "shape from x" problem. Compared to laser rangers, the performance of the CVL light-stripe range scanner is modest in range, accuracy and speed. But at a very low cost, it produces images which are comparable to laser images. On these images, software can be tested before it is ported to laser sensor systems. Here are a few areas in which the CVL light-stripe ranger makes experimentation possible:

Edge detection in range images
Edge preserving smoothing
Segmentation
Height maps
Dynamic (continuous) path planning
Motion estimate of the range sensor
Object recognition
Data fusion with video images
Detection of moving obstacles from sequences of range images.

The reader is refered to Asada [8] for examples of images of models of complex outdoor scenes produced by the CVL range scanner, and for tests on these images of several algorithms in the areas mentioned above.

22. Conclusions

We have described the CVL implementation of a light-stripe range scanner. We gave a precise and complete description of the hardware, interfaces and codes so that persons not involved in its development can use the system and understand it, and possibly find ways to improve on it. This scanner can produce images from models of complex natural scenes, for developing algorithms from low level processing to navigation. These solutions could be immediately applicable to robotic vehicles equipped with the small inexpensive laser rangers to come.

Acknowledgements

We would like to thank Minoru Asada for helpful discussions and for his creative use of the range scanner [8]. We also thank Pr. Azriel Rosenfeld for reviewing the manuscript.

References

[1] P. Veatch and L.S. Davis, "IRS: A Simulator for Autonomous Land Vehicle Navigation", University of Maryland, Center for Automation Research Technical Report 310, July 1987.

[2] M. Hersman, F. Goodwin, S. Kenyon, and A. Slotwinski, "Coherent Laser Radar Application to 3D Vision", Vision '87, Detroit, June 8-11, 1987.

[3] R.A. Jarvis, "A Perspective on Range Finding Techniques for Computer Vision", IEEE Trans. PAMI-5, pp.122-139, 1983.

[4] B.C. Bolles, J.H. Kremers and R.A.Cain, "A Simple Sensor to Gather Three-Dimensional Data", SRI Technical Note 249, July 1981.

[5] G.J. Agin, "Calibration and Use of a Light Stripe Range Sensor Mounted on the Hand of a Robot Arm", Carnegie-Mellon University Robotics Institute Technical Report 85-20, November 1985.

[6] Y. Shirai, "Recognition of Polyhedrons with a Range Finder", Pattern Recognition, vol.4, pp.243-250, 1972.

[7] K. Boyer and A.C.Kak, "Color-Encoded Structured Light for Rapid Active Ranging", IEEE Trans. PAMI-9, pp. 14-28, 1987.

[8] M. Asada, "Building a 3D World Model for a Mobile Robot from Sensory Data", University of Maryland, Center for Automation Research Technical Report 332, October 1987.

[9] O. Ozeki, T. Nakano, and S. Yamamoto,"Real-Time Range Measurement Device for Threee-Dimensional Object Recognition", IEEE Trans. PAMI-8, pp. 550-554, 1986.

[10] T. Lozano-Perez, J.L. Jones, E. Mazer, P.A. O'Donnell, and W.E.Grimson, "Handey: A Robot System that Recognizes, Plans, and Manipulates, 1987 Int. Conf. on Robotics and Automation, vol.2, pp. 843-849, March 1987.

[11] D.F. DeMenthon and Minoru Asada, "Calibration of a Light-Stripe Range Scanner", University of Maryland, Center for Automation Research Technical Report (in preparation).

APPENDIX A

Specifications for the equipment used in the Light-Stripe Range Scanner

Camera: *Sony*, CCD Model XC-38, with genlock unit CBK-38GL.

Lens system: *Switar*, focal length 10 mm.

Light Projector: *Dolan Jenner*, Model 180, 150 W.

Fiberoptics Light Guide: *Dolan Jenner*, SK3836, 4 ft, flaring to 2.5" x 0.125" line.

Stepper Motor: *SLO-SYN*, M061-FD02, 1.8 degree steps, rated at 5 percent accuracy.

Gear Box: *LINK* "HI-PRECISION", GP-122, Ratio 20:1, Precision Class 3 Gears.

Appendix B

Codes for the CVL Light-Stripe Ranger

```
program MIRTABLE (input, output);
{*****************************************************************

*****************************************************************}
{Daniel DeMenthon, Nov. 86}
{Revision August 87}
{Parameters changed from calibration, Sept. 87}
{Modified to include choice of table formats, Nov.87}
{$Q+}
{$s=500000}
{$f=0.&.GENERAL.SI}
 const
  pi = 3.14159;
  degtorad = 0.01745;
  radtodeg = 57.296;
  nypix = 512; (* no. of image pixels in y direction *)
  xpixo = 264; (* x of center of image in pixels *)
  ypixo = 238; (* y of center of image in pixels *)
  fx = 597;{from calibration at 3ft, diaphragm 22}
  fy = 758;
{Image plane of equiv. pin-hole camera is 30 mm in front of chip:}
  yfocmir = 171; (* in mm, dist between length center & mirror *)
  zfocmir = 75; (* in mm, dist along parall to base *)
  tilt = 0.0; (* tilt of camera w/ respect to base *)
  astep = 0.045; (* in deg, step angle of mirror*)
  nbsteps = 512;{nb. of mirror steps}
  firstbeta = 16.3; {angle beta for scan start of mirror}
  tablim = 16#880000; {image no.2, range table}
  tablenb = 1;{image no.2}
  nonlinear = 1;
  linear = 2;
type
  pixmem = word;
  pixptr = record
  case boolean of
    true: (p: ^pixmem);
    false:(i: integer);
  end;{pixptr}

  anglarray = array [1..nypix] of real;
  string64 = string[64];
  filename = string[80];

 var
  gamma: real;{angle between box axis and mirror-nodal pt line}
  OM: real;{segment between mirror & nodal pt}
  status : integer;
  tablenum : imagenum;
  myfile : filename;
  mypict : string64;
```

```
  minrange : real;
  maxrange : real;
  midrange: real;{range for which nonlinear grey is at inflexion point}
  mirstart: real;
  initalpha: real;
  tabletype: integer;
  goodanswer:boolean;

PROCEDURE initvdp(var status: integer); FORWARD;

procedure writeimg(var filedescr: filename;
                   dest: imagenum;
                   mode: integer;
                   oldflag: integer;
                   var picname: string64;
                   var status: integer); FORWARD;

 function getbeta (stepnb : integer) : real;
 {get angle between OM and plane of light}
  var
   mir, beta : real;
 begin
  mir := mirstart + (stepnb - 1) * astep;
  beta := 2 * mir - gamma;
  {writeln('beta ', beta);}
  getbeta := beta;
 end;{beta}

 function getdelta (pixrow : integer) : real;
 {get angle between camera axis and image row plane}
  var
   ypix : integer;
   delta : real;
 begin
  ypix := pixrow - ypixo;
  delta := radtodeg * arctan(ypix / fy);
  {writeln('delta ', delta);}
  getdelta := delta;
 end;{getdelta}

 function getalpha (pixrow : integer) : real;
 {get angle in deg. between OM and image row plane}
  var
   alpha : real;
 begin
  alpha := 90 + gamma + getdelta(pixrow);
  {writeln('alpha ', alpha);}
  getalpha := alpha;
 end;{getalpha}
```

```
 function getslowrange (pixrow : integer;
         mirstep : integer) : real;
 {range from mirror to light stripe}
  var
   alpha, beta, range, sinangle : real;
 begin
  beta := degtorad * getbeta(mirstep);
  alpha := degtorad * getalpha(pixrow);
  sinangle:= sin(alpha + beta);
  if (sinangle = 0.0) then
    range := -1
  else
    range := OM * sin(alpha) / sinangle;
  getslowrange := range;
 end;{getslowrange}

 function getrange (alpha: real;
                    mirstep: integer) : real;
 {range from mirror to light stripe}
 {use precalculated list of alphas}
  var
   beta, range, sinangle : real;
 begin
  beta := degtorad * getbeta(mirstep);
  alpha := degtorad * alpha;
  sinangle:= sin(alpha + beta);
  if (sinangle = 0.0) then
    range := -1
  else
    range := OM * sin(alpha) / sinangle;
  getrange := range;
 end;{getrange}
```

```
procedure getlist(var alphalist: anglarray);
{make an array of all the alpha angles in degrees}
 var
  irow: integer;
 begin
  for irow:= 1 to nypix do
    alphalist[irow] := getalpha(irow);
 end;{getlist}

 function nonlinrangetogrey (range : real) : word;
{give a grey level corresponding to range}
{coding by a S-curve}
  var
   grey : word;
   rangexp: real;
 begin
  rangexp := exp(-0.5 * sqr((range - minrange) / (midrange - minrange)));
```

```
  {rangexp is 1 for minrange, inflexes at midrange}
  grey := 16 * round(2047 * rangexp);
  {2047 is 7FF}
  {16 shifts the bits 4 bits, leaving 0 in the 4 graphic bits}
  nonlinrangetogrey := grey;
 end;
  (* end of function nonlinrangetogrey *)

 function linrangetogrey (range : real) : word;
{give a grey level corresponding to range}
{coding by a linear function}
  var
   grey : word;
   greylin : integer;
 begin
 { writeln('range ', range:8:3);}
  if (range>0) and (range <= maxrange) then
    greylin := trunc(maxrange - range)
  else
    greylin := 0;
{greylin is linear, decreases from  255 when range = 193}
{to zero when  range = maxrange, maxrange = 448}
  grey := 256 * greylin;
 {writeln('greylin, grey ', greylin:7,  grey:10);}
{256 shifts the 8 bits of round(greylin) of 8 bits}
{putting them in the top 8 bits of a word}
  linrangetogrey := grey;
 end;{linrangetogrey}

 procedure maketable;
{Fill up an image with grey levels representing possible ranges}
{for each mirror step, 1 per line, and each pixel row, 1 per column}
  var
   mirstep, pixrow : integer;
   alphalist: anglarray;
   alpha, thisrange: real;
   greylevel: word;
   tablimage : pixptr;
 begin
  getlist(alphalist);{list of angles alpha for every pixel row}
  tablimage.i := tablim;{starting address of tablimage}
  for mirstep := 1 to nbsteps do
   begin
     writeln('mirstep= ',mirstep);
     for pixrow := 1 to nypix do
      begin
       {writeln('pixrow= ', pixrow);}
       alpha:= alphalist[pixrow];
       thisrange:= getrange(alpha, mirstep);
       {if pixrow = 1 then}
```

```
          {writeln('range ', thisrange);}
        if (tabletype = nonlinear) then
          greylevel := nonlinrangetogrey(thisrange)
        else if (tabletype = linear) then
          greylevel := linrangetogrey(thisrange);
        {writeln('greylevel ', greylevel);}
        tablimage.p^ := greylevel;
        tablimage.i := tablimage.i + 2;
      end; (* for pixrow *)
    end; (* for mirstep *)
 end; (* maketable *)

procedure initvalues;
begin
 OM := sqrt(yfocmir * yfocmir + zfocmir * zfocmir);
 gamma := radtodeg * arctan(zfocmir / yfocmir);
 mirstart := (firstbeta + gamma)/2;{starting angle for mirror}
 minrange := getslowrange(1, 1);{pixrow=1, mirstep=1}
 {minrange is minimum detected range}
 writeln('minrange ', minrange);
 maxrange := minrange + 255;
end;{initvalues}

begin (* main *)

 initvalues;
 initvdp(status);{initialize video display}
 if status<>0 then
    writeln('initvdp error ');
 repeat
  writeln('type 1 for a nonlinear range coding, 2 for a linear range coding ');
   readln(tabletype);
   goodanswer:=(tabletype=nonlinear) or (tabletype=linear);
   if not goodanswer then
      writeln('wrong answer ');
 until goodanswer;
 if (tabletype = nonlinear) then
   begin
   writeln('input midrange, distance in mm to middle of scene; remember it');
   readln(midrange);
   myfile:= 'rgtable.im';{where table is going to be stored}
 end{if}
 else if (tabletype = linear) then
   myfile:= 'rgtab2.im';
maketable;
{Store image of table to disk:}
 tablenum := tablenb;
 mypict:= ' ';
 writeimg(myfile, tablenum, 1, 2, mypict, status);
end. (* end of main *)
```

```
program MIRANGE(input, output);

{*****************************************************************}
{Nov. 86, revisions July 87, Nov. 87}
{Get a range matrix from the mirror point of view}
{Put the range lookup table in image no.2}
{Get an image with no stripe}
{Get images with stripes}
{Send to THIN procedure}
{*****************************************************************}

{$Q+}
{$s=500000}
{$f=0.&.GENERAL.SI}

CONST
    pi = 3.14159;
    degtorad = 0.01745;
    radtodeg = 57.296;
    ndelay = 100;{for delay loop between steps, minimum is 30}
    fullturnsteps = 8000;{nb of steps for 360 deg. turn}
    astep = 0.045;{angle for 1 strep, in deg.}
    nbsteps = 512;
    yfocmir = 171; {in mm, dist. from optical axis to mirror}
    zfocmir = 75; {in mm, dist. from box axis to lens center}
    rangemap = 0; {image no.1, matrix of distances as seen from mirror}
    rangestart = 16#800000;{starting address of range image}
    tablenb = 1;{image no.2, table of distances, one column per step nb}
    tablestart = 16#880000;{starting address of table}
    nolitnb = 2; {image no.3, no stripe}
    nolight = 16#900000; {image no.3, no stripe}
    pluslitnb = 3;{image no.4, with stripe}
    pluslight = 16#980000; {image no.4, with stripe}
    camrange = 4;{image no. 5, range map from camera}
    camstart = 16#A00000;
    green = 2;
    greennb = 16386; {16384 + 2}
    greenstart = 16#500000;
    firstbeta = 16.3; {angle beta for scan start of mirror}
    nonlinear = 1;
    linear = 2;
TYPE
    string64 = string[64];
    filename = string[80];

VAR
  status : integer;
  tablenum : imagenum;
  myfile : filename;
  mypict : string64;
```

```
  answer: char;
  imirstart: integer;
  threshlevel : real;
  thisthresh: word;
  gamma : real;
  tabletype: integer;
  goodanswer:boolean;

PROCEDURE initvdp(var status: integer); FORWARD;

PROCEDURE digitize(dsnum, channels, upper: integer;
                   dest1, dest2, dest3: imagenum;
                   imgdig: integer;
                   var status: integer); FORWARD;

PROCEDURE subi(s1, s2, dest: imagenum;
                scale: Boolean;
                var status: integer); FORWARD;

PROCEDURE lshift(src, dest: imagenum;
                count: word;
               var status: integer); FORWARD;

PROCEDURE extrema(src: imagenum; var minval,maxval: real;
                var minrow, mincol: coordinate;
                var maxrow, maxcol: coordinate;
                var status: integer); FORWARD;

FUNCTION map: integer; FORWARD; {initializes TK's step routine}

PROCEDURE step; FORWARD; {TK's step routine}

procedure thin(table, nostripe, striped: integer;
               imir: integer;
               thresh: word;
               result1, result2: integer); FORWARD;{TK's thinning routine}

procedure readimg(var filedescr: filename;
                  dest: imagenum;
                  mode: integer;
                  oldflag: integer;
                  var picname: string64;
                  var status: integer); FORWARD;

PROCEDURE scanmirror;
{control mirror}
VAR
  status, imirstep, idelay: integer;
BEGIN
{get picture with  no stripe, when mirror at home position:}
```

```
  writeln('digitize home image ');
  digitize(0, green, 1, nolitnb, nolitnb, nolitnb, 1, status);
  lshift(nolitnb, nolitnb, 1, status);
{go to initial position of mirror:}
  FOR imirstep:= 1 to imirstart DO
  BEGIN
    step;
    FOR idelay := 1 to ndelay DO
      ;
  END;{to start position}
{get picture at first position, imirstart}
  writeln('digitize image for imirstart');
  digitize(0, green, 1, pluslitnb, pluslitnb, pluslitnb, 1, status);
  lshift(pluslitnb, pluslitnb, 1, status);{get bit out of sign bit}
{to be sure that mirror turned & still when digitizing, process previous
 digitizing when mirror is getting positionned:}
  writeln('loop on mirror positions ');
  FOR imirstep:= 1 to nbsteps DO
  BEGIN
    step;
    writeln('steps from home ', (fullturnsteps - imirstart - imirstep));
 {put thinned stripe into range images from camera & mirror views:}
    thin(tablestart, nolight, pluslight,
         imirstep, thisthresh, camstart, rangestart);
 {digitize new image:}
    digitize(0, green, 1, pluslitnb, pluslitnb, pluslitnb, 1, status);
    IF status<>0 then
      writeln('scanrange digitize error');
    lshift(pluslitnb, pluslitnb, 1, status);{get bit out of sign bit}
      {Note: last step is not processed, but still 512 steps have been}
  END;{for}
  {bring mirror home, completing the circle:}
  FOR imirstep:= 1 to (fullturnsteps - imirstart - nbsteps) DO
  BEGIN
    step;
    FOR idelay := 1 to ndelay DO
      ;
  END;{to start position}
END;{scanmirror}

procedure initialize;
var
 status: integer;
begin
  {angle from box axis to mirror-to-lens line:}
  {nb of steps for mirror to go from home to starting position:}
  gamma := radtodeg * arctan(zfocmir / yfocmir);
  imirstart := round((firstbeta + gamma) /(2 * astep));
  {set range image to zero:}
  subi(rangemap, rangemap, rangemap, false, status);
```

```
  subi(camrange, camrange, camrange, false, status);
  initvdp(status);{initialize video display}
  if status<>0 then
    writeln('initvdp error ');
  {initialize vicom-stepper connection:}
  status:= map;
  if status<>0 then
    writeln('map error');
end; {initialize}

BEGIN {main}
  initialize;
  tablenum := tablenb;
   repeat
  writeln('type 1 for a nonlinear range coding, 2 for a linear range coding ');
   readln(tabletype);
   goodanswer:=(tabletype=nonlinear) or (tabletype=linear);
   if not goodanswer then
      writeln('wrong answer ');
 until goodanswer;
 if (tabletype = nonlinear) then
   myfile:= 'rgtable.im'
  else if (tabletype = linear) then
     myfile:= 'rgtab2.im';
  mypict:= ' ';
  readimg(myfile, tablenum, 1, 2, mypict, status);
  end;{if}
  writeln('input threshold, between 0 and 1');
  readln(threshlevel);
  thisthresh:= round(threshlevel * 32768);{thisthresh is word}
 {rotate mirror, get striped images & fill range map:}
  scanmirror;
END.
```

```
          SECTION 9
*
          XDEF      MAP,STEP,PUT,GET
*
REGS      REG       D0-D7/A0-A6
*
ODDADDR   EQU       $7DE001
EVENADDR  EQU       $7DE000
*
* PARAMETER BLOCK FOR GETSEG TRAP
*
TSK_NAM   DC.L      0                   TASK NAME
SES_NUM   DC.L      0                   SESSION NAME
GET_OPT   DC.W      $100                OPTIONS
SEG_ATT   DC.W      $800                SEGMENT ATTRIBUTES
SEG_NAM   DC.L      'SEG3'              SEGMENT NAME
SEG_ADD   DC.L      $200000             START ADDRESS
SET_LEN   DC.L      $DFFFFF             SEGMENT LENGTH
*
STATUS    DC.B      0                   STATUS FLAG
          DC.B      0
*
*
*     THIS SUBROUTINE MAPS THE IMAGE MEMORY AND CONTROL REGISTERS
*          INTO THE ADDRESS SPACE OF THE PROGRAM
*
MAP       MOVEM.L   REGS,-(SP)          SAVE THE WORLD
          LEA       TSK_NAM,A0          GET ADDRESS OF GETSEG BLOCK
          MOVE.L    #$0001,D0           SET DIRECTIVE TO GETSEG
          TRAP      #1                  EXECUTE DIRECTIVE
          BNE.S     GSG_ERR             IF ERROR SET STATUS
          CLR.L     64(SP)              SET STATUS TO OK
          BRA.S     GSG_RTN             EXIT
GSG_ERR   MOVE.B    #$20,67(SP)         SET GETSEG ERROR CODE
GSG_RTN   MOVEM.L   (SP)+,REGS          RESTORE WORLD
          RTS                           RETURN
*
*     THIS SUBROUTINE STEPS THE MOTOR ONCE
*
STEP      MOVE.B    #00,EVENADDR
          RTS

*
*     THIS SUBROUTINE PUTS DATA INTO TO THE BUFFER
*
PUT       MOVE.B    7(SP),ODDADDR
          RTS
*
*     THIS SUBROUTINE GETS DATA FROM THE BUFFER
*
GET       MOVE.B    ODDADDR,7(SP)
          RTS

          END
```

Pseudocode for the THIN procedure

```
Save all registers except A7 (stack pointer) on stack
get parameters from stack

set A1 = row address of range information for current stripe , specified by IMIR
set A6 = row address of mirror result image
set A5 = start address of row-start array

if starting scan at bottom of image then
      fill row-start array with offsets from the top of the image to the
      bottom of the image for each column
endif

for all the columns of the image do
      set D6 = row-start offset for this column
      goto xx1

/*
    Find the bottom of the stripe by looking for the first pixel whose value
    is above the treshold
*/

    xx0:        D6 = D6 - rowlength (1024 bytes -> 512 cols at 2 bytes per pixel)
                if D6 < top of image goto xx4

    xx1:        set D0 =  pixel at offset D6 from stripe image
                subtract from D0 pixel at offset D6 from image without stripe
                if  D0 < THRESH goto xx0

          save D6 in row-start array at present column position
          set D5 = D6

/*
    Find the top of the stripe by looking for the first pixel whose value
    is below the threshold
*/

    xx2:        D5 = D5 - rowlength
```

```
                if D5 < top of image goto xx5

                set D0 =  pixel at offset D5 from stripe image
            subtract from D0 pixel at offset D5 from image without stripe
                if  D0 > THRESH goto xx1

            set D4 = D5
            set D3 = MINZER (minimum number of zeros)
/*
    Check for at least two zero pixels above the proposed top of stripe.
    If there are fewer than two zeros above the line then go back and look
    for a new top of line.
*/

    xx2.1:      D4 = D4 - rowlength
                if D4 < top of image goto xx3

                set D0 =  pixel at offset D4 from stripe image
            subtract from D0 pixel at offset D4 from image without stripe

                if DO > THRESH then
                    set D5 = D4
                    goto xx2
                else
                    D3 = D3 -1
                    if D3 > -1 goto xx2.1
                endif

/*
    We now have both the top and bottom of the stripe.  Now check to see
    if the line is thick enough. If it is not thick enough then go back to
    the top of the loop and try to find the real line starting from the
    row we thought was the top of the stripe
*/

    xx3:    set D6 = D6 - D5   /* get number of rows between top and
                                  bottom of stripe */
            if D6 < MINROWS*rowlength    /* is stripe thick enough */
                set D6 = D5
```

```
                goto xx1
            endif

/*
    Compute the offset of the row half way in between the top and bottom
    rows of the stripe.  Then put the range values of this mirror angle
    and row into the resulting images.
*/

    xx3.5:  set D6 = (D6 + 1) / 2 + D5 /* compute mid-row of stripe */
            set D6 = D6 and 0xFC00  /* keep only the row number info */
            set D4 = D6 / 512
            set pixel at offset D6 form A2 = pixel at offset D4 from A1
            set pixel at A6 = pixel at offset D4 from A1

/*
    Increment the pointers to the next column.
*/
    xx4:
            set A5 = A5 + 4    /* row-start pointer */
            set A4 = A4 + 2    /* image without stripe pointer */
            set A3 = A3 + 2    /* image with stripe pointer */
            set A2 = A2 + 2    /* camera result pointer */
            set A6 = A6 + 2    /* mirror result pointer */
        endfor

        restore registers from stack
        return to calling program

    xx5:    set D5 = 0
            if D6 < MINROWS*rowlength goto xx4
            goto xx3.5

    END
```

```
*
        XDEF    THIN
*
REGS    REG     D0-D7/A0-A6
*
RANGE   EQU     26      RANGE DATA(N MIRROR ANGLE BY 512 ROWS OF IMAGE)
IMAGE   EQU     22      IMAGE WITHOUT STRIPE
STRIPE  EQU     18      IMAGE WITH STRIPE
IMIR    EQU     14      MIRROR ANGLE (1-N)
THRESH  EQU     12      THRESHOLD VALUE
RESULT  EQU     8       RESULT OF PROCESS
RESULT2 EQU     4       RESULT OF PROCESS
*
MINROWS EQU     2
MINZER  EQU     1
*
*
        SECTION 7
*
*    STATIC VARIABLES....
*
ROW     DS.L    512
*
*
        SECTION 9
*
*    PROGRAM SPACE
*
THIN    MOVEM.L REGS,-(SP)
        LEA     ROW,A5                  ADDRESS OF LAST ROW TABLE
        MOVE.L  IMAGE+60(A7),A4         ADDRESS OF IMAGE WITHOUT STRIPE
        MOVE.L  STRIPE+60(A7),A3        ADDRESS OF STRIPE IMAGE
        MOVE.L  RESULT+60(A7),A2        ADDRESS OF RESULT IMAGE
        MOVE.L  RESULT2+60(A7),A6       ADDRESS OF RESULT IMAGE
        MOVE.L  RANGE+60(A7),A1         ADDRESS PIXEL DISTANCE TABLE
        MOVE.W  THRESH+60(A7),D1        THRESHOLD VALUE
*
*
        MOVE.L  IMIR+60(A7),D0          ARE WE STARTING THE SCAN (AT BOTTOM)
        SUBQ.L  #1,D0
        ASL.L   #8,D0
        ASL.L   #2,D0
        ADD.L   D0,A1
        MOVE.L  #511*1024,D2
        SUB.L   D0,D2
        ADD.L   D2,A6
        TST.L   D0
        BNE.S   LLL                     NO, PROCEED NORMALLY
*
*
*    INITIALIZE THE LAST ROW ARRAY
*
        MOVE.L  A5,-(A7)                MOVE TO WORK REG
        MOVE.W  #511,D2                 LOOP FOR 512 COLS
        MOVE.L  #481*1024,D0            OFFSET TO BOTTOM ROW OF IMAGE
KKK     MOVE.L  D0,(A5)+                STORE IN LAST ROW LIST
        DBRA    D2,KKK                  LOOP FOR ALL 512 COLS
        MOVE.L  (A7)+,A5
*
*    PROCESS IMAGE
```

```
*
LLL     MOVE.W  #511,D2
XXX     MOVE.L  (A5),D6                 BOTTOM ROW # OF LAST LINE IN THIS COL.
        BRA.S   XX1
*
XX0     SUB.L   #1024,D6                        UP ONE ROW
        BMI.S   XX4                     BACKED UP PASSED TOP OF SCREEN,
*                                       TERMINATE THIS PASS
XX1     MOVE.W  (A3,D6.L),D0            IS PIXEL 0?
        SUB.W   (A4,D6.L),D0
        CMP.W   D1,D0
        BLT.S   XX0                     YES, MOVE UP ONE ROW AND TRY AGAIN
*
        MOVE.L  D6,(A5)                 SAVE BOTTOM OF LINE IN ARRAY
        MOVE.L  D6,D5                   SAVE BOTTOM OF LINE
XX2     SUB.L   #1024,D5                        UP ONE ROW
        BMI.S   XX5                     BACKED UP PASSED TOP OF SCREEN, Fix
        MOVE.W  (A3,D5.L),D0                    IS PIXEL 1?
        SUB.W   (A4,D5.L),D0
        CMP.W   D1,D0
        BGE.S   XX2                     YES, MOVE UP ONE ROW
*
        MOVE.L  D5,D4
        MOVE.W  #MINZER,D3
XX2.1   SUB.L   #1024,D4
        BMI.S   XX3
        MOVE.W  (A3,D4.L),D0
        SUB.W   (A4,D4.L),D0
        CMP.W   D1,D0
        DBGE    D3,XX2.1
        BLT.S   XX3
        MOVE.L  D4,D5
        BRA.S   XX2
*
XX3     SUB.L   D5,D6                   DIFFERANCE BETWEEN TOP AND BOTTOM
        CMP.L   #MINROWS*1024,D6        HAVE MORE THAN MIN # OF ROWS?
        BGE.S   XX3.5                   YES, CONTINUE...
        MOVE.L  D5,D6                   NO, START FROM WHERE WE LEFT OFF
        BRA.S   XX1
XX3.5   ADD.L   #1024,D6                        TO ROUND UP
        LSR.L   #1,D6                   DIVIDE BY 2
        AND.W   #$FC00,D6               KEEP ONLY ROW INFO
        ADD.L   D5,D6
        MOVE.L  D6,D4
        LSR.L   #8,D4
        LSR.L   #1,D4
        MOVE.W  (A1,D4.L),(A2,D6.L)             PUT DIST. TO PIX IN RESULT
        MOVE.W  (A1,D4.L),(A6)
*
*
XX4     ADDQ.L  #4,A5
        ADDQ.L  #2,A4
        ADDQ.L  #2,A3
        ADDQ.L  #2,A2
        ADDQ.L  #2,A6
        DBRA    D2,XXX
        MOVEM.L (SP)+,REGS
        RTS
*
*     MOVED PASSED THE TOP OF THE SCREEN SO RESET TO TOP OF SCREEN
```

```
*
XX5     CLR.L   D5                      SET TO TOP OF SCREEN
        SUB.L   D5,D6
        CMP.L   #MINROWS*1024,D6
        BGE.S   XX3.5                   CONTINUE
        BRA     XX4
*
*
        END
```

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS N/A
2a. SECURITY CLASSIFICATION AUTHORITY N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) CAR-TR-337 CS-TR-1962	5. MONITORING ORGANIZATION REPORT NUMBER(S) N/A
6a. NAME OF PERFORMING ORGANIZATION University of Maryland 6b. OFFICE SYMBOL (If applicable) N/A	7a. NAME OF MONITORING ORGANIZATION U.S. Army Engineer Topographic Laboratories
6c. ADDRESS (City, State, and ZIP Code) Center for Automation Research College Park, MD 20742-3411	7b. ADDRESS (City, State, and ZIP Code) Fort Belvoir, VA 22060-5546
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Defense Advanced Research Projects Agency 8b. OFFICE SYMBOL (If applicable) ISTO	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DACA76-84-C-0004
8c. ADDRESS (City, State, and ZIP Code) 1400 Wilson Blvd. Arlington, VA 22209	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO.: 62301E PROJECT NO.: TASK NO.: WORK UNIT ACCESSION NO.:

11. TITLE (Include Security Classification)

Production of Dense Range Images with the CVL Light-Stripe Range Scanner

12. PERSONAL AUTHOR(S)
Daniel DeMenthon, Tharakesh Siddalingaiah and Larry S. Davis

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM _____ TO N/A	December 1987	52

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report describes a system able to produce 512 x 512 range images of model scenes in the laboratory. This ranging instrument, which comprises a light-emitting slit, a cylindrical lens, a step-motor controlled mirror and a CCD camera, is compact enough to be mounted on the tool plate of a robot arm. The light source itself is mounted away from this structure, and the light is brought to the slit by a flexible fiberoptic light guide. The robot arm's motion can be controlled by inputs from the range scanner, for simulation of autonomous vehicles equipped with rangers. This system is programmed to produce range images which are comparable in many respects to range images produced by laser range scanners. With this similitude of formats, software for edge detection, object recognition, dynamic path planning or data fusion with video images can be developed on range images produced by this laboratory equipment and can be easily ported to laser ranging systems.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL Linda Graff	22b. TELEPHONE (Include Area Code) (202) 355-2736	22c. OFFICE SYMBOL ETL-RI-T

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED