Robot Spatial Perception by Stereoscopic Vision and 3D Evidence Grids

Hans P. Moravec

CMU-RI-TR-96-34

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

September 1996

Portions of this research were sponsored by the Office of Naval Research under contract number N0094-93-1-0765, by Daimler Benz Research, Berlin, by Thinking Machines Corp., and by the CMU Robotics Institute

Acknowledgments:

This work was done while the author was on a six month sabbatical visit. Thanks to my hosts Frieder Lohnert, Dimitris Georgianis, Volker Hansen and the other members of Daimler Benz Research, Berlin, who provided the encouragement and means to complete this work. Special thanks go to Ralph Sasse for several times volunteering critical help, taking time from his own consuming research. Thanks also to my CMU support network, especially Martin Martin, Mike Blackwell and Kevin Dowling. A major component of the program described here, the evidence ray thrower, was written during my 1992 sabbatical year at Thinking Machines Corp. in Cambridge, Massachusetts - many thanks for that opportunity. Long term support for this research was provided since 1982 by the US Office of Naval Research under N00014-93-1-0765 and previous contracts. The impetus for the evidence grid idea came from a research contract from Denning Mobile Robotics Inc. in 1983. Final thanks go to Christopher Priem for so carefully studying my program, and generating the large data set whose images are found in the report's appendix.

Robot Spatial Perception by Stereoscopic Vision and 3D Evidence Grids

Abstract

Very encouraging results have been obtained from a new program that derives a dense three-dimensional evidence grid representation of a robot's surroundings from wide-angle stereoscopic images. The program adds several spatial rays of evidence to a grid for each of about 2,500 local image features chosen per stereo pair. It was used to construct a 256x256x64 grid, representing 6 by 6 by 2 meters, from a hand-collected test set of twenty stereo image pairs of an office scene. Fifty nine stereo pairs of an 8 by 8 meter laboratory were also processed. The positive (probably occupied) cells of the grids, viewed in perspective, resemble dollhouse scenes. Details as small as the curvature of chair armrests are discernible. The processing time, on a 100 MIPS Sparc 20, is less than five seconds per stereo pair, and total memory is under 16 megabytes. The results seem abundantly adequate for very reliable navigation of freely roaming mobile robots, and plausibly adequate for shape identification of objects bigger than 10 centimeters. The program is a first proof of concept, and awaits optimizations, enhancements, variations, extensions and applications.

1 Introduction

This report describes a new program that transforms stereoscopic images, as could be obtained from a camera-equipped roving robot, into dense three-dimensional maps of the robot's immediate surroundings. The program generates 100 to 1,000 times as much good map data as previous systems relying on 2D or sparse 3D representations, suggesting that the statistical reliability of navigation programs using on it should be very high. The program was tested with 20 stereo image pairs of an office, and a second data set of 59 stereo pairs of a larger laboratory. The resulting 3D maps show object details to a scale of about 10 cm. By using higher resolution grids, features of a few centimeters can be resolved.

Before mapping, the stereo camera fixture is calibrated. The fixture is leveled, and positioned, with reasonable accuracy, perpendicularly a known distance in front of a screen patterned with a precise square array of about 400 black spots on a white background. A program, designed to deal with the fish-eye distortion of wide angle lenses, finds the spots in the cameras' digitized images, and constructs a *rectification* function for each camera that corrects for the distortion and small mounting misalignments. In use, the rectification functions are compiled into image-sized lookup tables, which geometrically transform raw camera images so they appear to have come from a specified ideal optical geometry.

During mapping, a sequence of stereo image pairs is processed. The results accumulate in a three-dimensional array called an *evidence grid*, whose cells represent regions of space. Each grid cell accumulates evidence, positive and negative, that its region is occupied. The grid cells are initialized to zero, indicating no evidence for or against the cell being occupied. After sufficient data has accumulated, blocks of negative cells indicate free space, while positive cells define objects. The program's computationally efficient additive "weight of evidence" metric can be interpreted in probability theory as Bayesian combination using a $\log\frac{p}{1-p}$ "log-odds" representation of probability.

Processing of each stereo pair begins with image rectification. An *interest operator* then selects about 2,500 local features in the two images, each chosen to be a good candidate for locating in the other image of the pair. For each feature, a *correlator* scans the possible corresponding locations in the other image, producing a curve of goodness of match. The peaks in this curve each correspond to a possible distance for the feature. For each of the possibilities, the program throws two rays of evidence into the grid, one from each camera position towards the inferred feature position. The rays have positive evidence of occupancy at the feature, and negative evidence between the camera and feature positions, and are weighted for the estimated overall probability of correctness of each peak.

Our main example used a grid 256 cells wide by 256 cells deep by 64 cells high, scaled to represent a volume 6 meters by 6 meters by 2 meters. The 40 images in the data set were obtained by moving the tripod-mounted stereo fixture, tilted down about 15 degrees, by hand, to vertices on a 50 cm floor grid, maintaining parallel camera orientations. The resulting evidence grid was displayed in multiple 2D X, Y and Z slices, with probabilities shown in gray scale. A more wholistic viewing method generated perspective 3D images of the positive cells of the grid, with cells in about a dozen box-shaped regions, containing major objects, distinctively colored. The latter display clearly shows the shape of furniture sized objects in the scene, with a level of detail reminiscent of doll-house pictures.

Results from a second, larger, dataset of 59 stereo pairs of an 8 by 8 meter laboratory are shown in the appendix.

The existing program succeeds as a proof of concept, but awaits improvement in nearly every detail. Some of the work will likely be accomplished by a learning procedure, that adjusts many program parameters to maximize a quality criterion. A promising criterion is a comparison of some of the original images of the scene with corresponding “virtual” images of the grid. The grid will first be colorized in original scene colors, by back-projecting other original images into the grid’s occupied cells.

2 Research Background

The results described here build on 25 years of work in mobile robot perception. From 1971 to 1983 the author and colleagues developed stereo-vision-guided robots that drove through clutter by tracking a few dozen features, presumed to be parts of objects, in camera images. Through several versions, the control program gained speed and accuracy, but a brittle failure mode persisted, misdirecting the robot after approximately 100 meters of travel, when chance clusters of tracking errors fooled geometric consistency checks [Moravec83].

We invented the so-called evidence grid approach in 1983, to handle data from inexpensive Polaroid sonar devices, whose wide beams leave angular position ambiguous. Instead of determining the location of objects, the grid method accumulated the “objectness” of locations, arranged in a grid, slowly resolving ambiguities about which grid cells were empty or filled, as the robot moved. The first implementation worked surprisingly well, showing none of the brittleness of the old approach. It could repeatedly map and guide a robot across a cluttered test lab [Moravec-Elfes85]. It worked on a tree-lined path, in a coal mine, with stereo vision range data, combined stereo and sonar, and with probability theory replacing an ad-hoc formulation. Its first failure, in uncluttered surroundings with smooth walls, led us to a major extension, the learning of sensor evidence models. The evidence contributed by individual sonar readings had originally been hand-derived from the sensor’s signal pattern, a poor model for interaction with mirrorlike walls. We are now able to train the program to work nicely in mirror surroundings, and superbly elsewhere [Moravec-Blackwell93]. A summary of the work to date is found in [Martin-Moravec96].

Past work was with 2D grids of a few thousand cells, all that 1980s computers could handle in near real time. In 1992 we wrote a very efficient implementation of the central operation for three dimen-

sions, that can throw thousands of evidence rays per second into 3D grids with several million cells, on 100 MIPS computers. The work described in this report combines that program with several new components into a complete robot stereoscopic 3D mapping package.

3 Camera Calibration and Image Rectification

The **flatfish** calibration program assumes its images come from solid state cameras with geometrically precise imaging surfaces, and good quality optics whose distortions are rotationally symmetric about their optical axis. The program constructs rectification functions that transform slightly off-center, rotated fish-eye images from imperfectly mounted wide-angle cameras into images exhibiting specified simple projective geometry, suitable for direct use by stereoscopic vision programs. Though designed for wide-angle optics, it also works with narrow angle lenses exhibiting little distortion.

For each camera, the program inputs an image of a calibration spot pattern, an aim point on the pattern, the measured distance between the lens center and the pattern, and the desired angular field of view of the rectified image. It produces a file of calibration parameters which contains the necessary information to rectify images from the same camera to specifications. The program also contains code which implements the rectification. A version of this rectification code is packaged in **fisheye**, a set of image processing routines, where it is used to compile a lookup table for rapidly applying the rectification. A side effect of running **flatfish** is a trace file and about ten black and white and color images showing its working stages.

The calibration pattern is a wall-mounted square array of dark spots on a light background (figure 1), with one additional spot at a half-grid position at the center, serving as an absolute position reference. We have used dark magnetic spots on a steel whiteboard, and also a 2 meter by 1.5 meter computer printout. The program expects the pattern to be 15 to 35 spot spacings wide, and the spot spacing to be about twice the spot diameter, with wide tolerances. The expectations are defined by program parameters.

The calibration spot input image is assumed to come from a moderately accurate physical setup, with the camera line of sight as perpendicular as possible to the plane of the spot pattern arrayed in front of it, and its pixel axes nearly parallel to the rows and columns of the spot array. The program uses first-order corrections to remove the remaining angular errors in azimuth, elevation and roll, as well as scaling and recentering the result to the desired field of view, after removing radial fish eye distortion and non-unity aspect ratio. To calibrate a pair of stereo cameras, both cameras, held in their stereoscopic mounting configuration, are aimed at the same calibration pattern, as in figure 1. The flatfish program is run for each resulting image, with only the aim point on the calibration pattern specified differently for the two calibrations. The aim point for each camera defines the rectified view direction, and the separation of the two aim points on the pattern is specified equal to the physical separation of the cameras.

Flatfish begins by applying a *spot operator* to its image. At each pixel position this operator weighs hypotheses that there is a spot of a range of radii centered at the pixel. For each radius hypothesis, the operator calculates the means and variances of the intensities of the spot interior and an annular surrounding region. The score for the hypothesis is the outer minus the inner mean, minus fractions of the variances. The variance fractions define the program's tolerance for non-uniform lighting and glare in the background and spot areas. The "spotness" value for the pixel is the maximum score for all the radius hypotheses. That spotness value, and the corresponding radius, are recorded for each pixel. The program applies the operator up to the edge of the image by giving zero weight to portions of the spot mask that lie beyond the image boundaries. For a wide range of settings of the spot size range and variance fractions, the operator is positive in small central areas of all spots in a calibration

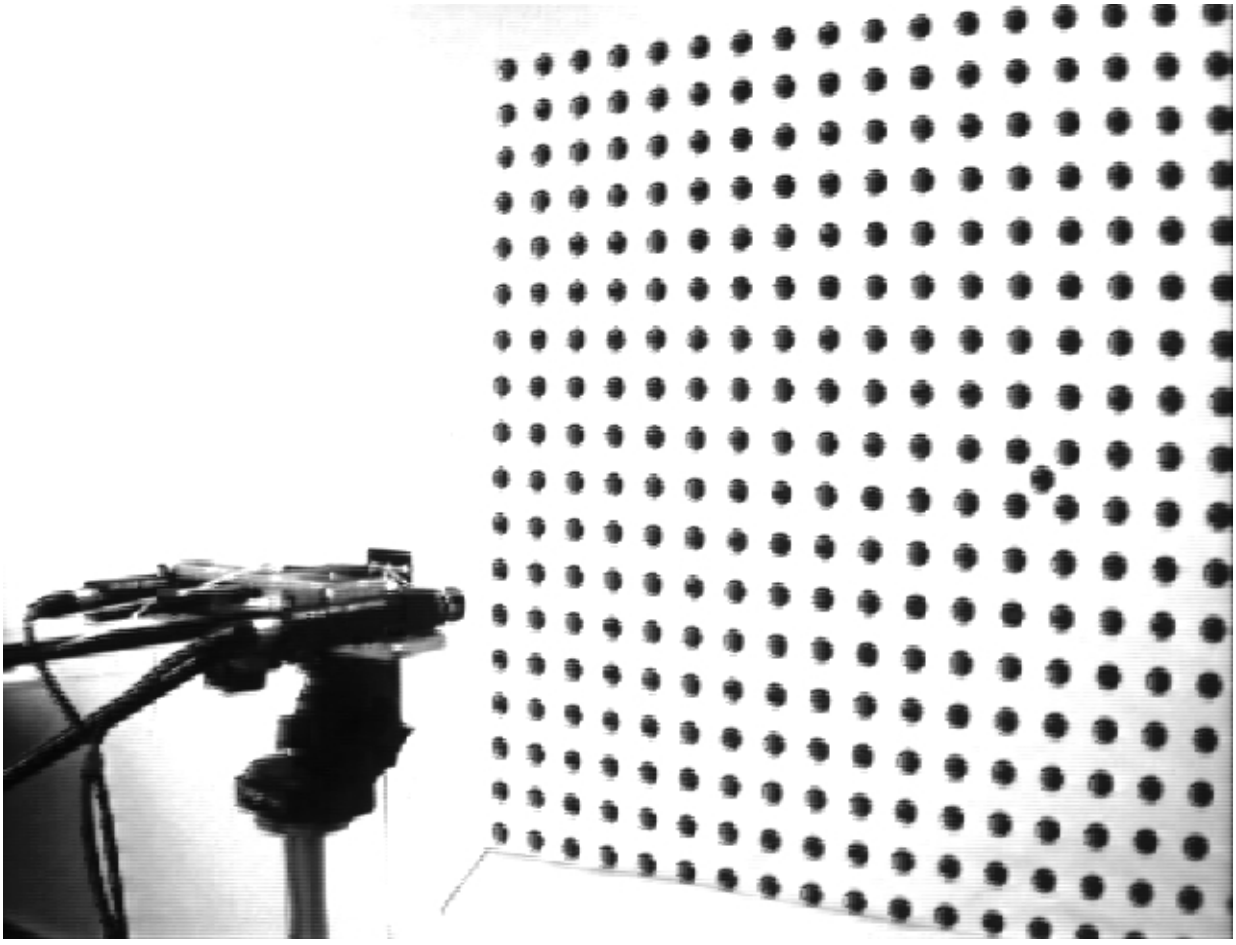


FIGURE 1 *A stereo camera assembly being calibrated.*

image, and negative elsewhere. It is usually negative everywhere when applied to general scenes not containing properly sized spot patterns. It is a very trustworthy operator.

After calculating spotness at every pixel, the program catalogs local maximum spot values and the corresponding spot radius, thus identifying each spot in the image. It finds the central spot in the pattern by noting which is closest to four other spots, relative to their spot sizes. It links the remaining spots into a grid, starting with a spot near the middle of the image, and using a rough idea of the spot spacing based on spot size to repeatedly find horizontally and vertically adjacent neighbors. At each step it uses the size and direction of the previous steps to adjust its rough idea of the local spot spacing and orientation. By this means it tracks the significant scale and orientation distortions found in the edges and corners of wide-angle images. The program assigns [row, column] coordinates to spots. The central spot has coordinates [0,0], the spot to its lower right is [0.5, 0.5], the spot below that is [1.5, 0.5], and so on. Via interpolation, each pixel in the image is then identified both by these "spot coordinates" as well as its original pixel coordinates.

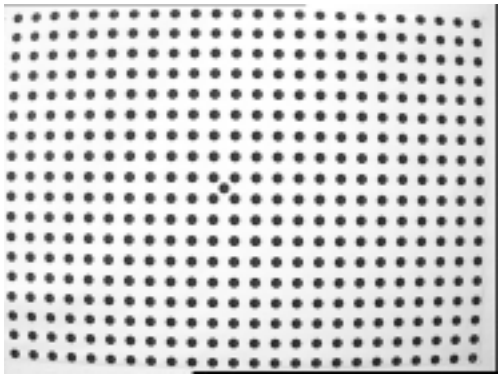
The program locates the optical axis in the image by finding the image location OA (and incidentally also the aspect ratio adjustment) that minimizes the scatter when the radial distance of each spot from

OA, measured in pixel coordinates, is plotted against the same radial distance measured in spot coordinates. The shape of the spot versus pixel radius curve characterizes the fish-eye distortion, and is represented in the program by a least-squares best-fit polynomial. With a 90 degree field of view lens (figure 3), we found that the pixel coordinates of the optimum OA remained stable within a few hundredths of a pixel as the camera was moved to various positions in front of a calibration grid. The root-mean-square deviation of the pixel versus spot radius scatter diagram, when OA was chosen randomly near the center of the image was about 2.5 pixels. The rms error was reduced to less than 0.5 pixels at the optimum OA.

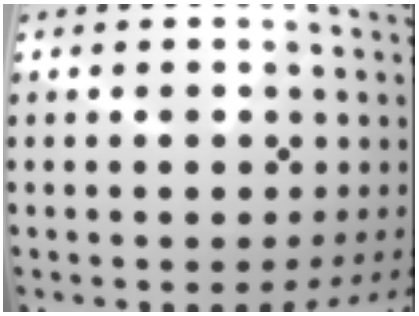
In another least-squares step, the program then finds and applies the image rotation (roll) that most nearly makes the fish-eye corrected spot pattern line up with the pixel rows and columns. It also shifts and scales the image to bring the requested aim point to the exact center, and to provide exactly the requested field of view across the width of the image raster. The largest source of error is probably in the user's estimate of the distance between the lens center and the test pattern, which affects the true angle of view. We use the lens iris ring position as an estimate for the lens center. The uncertainty diminishes as the pattern and the camera distance are made larger, or the lens is made smaller.

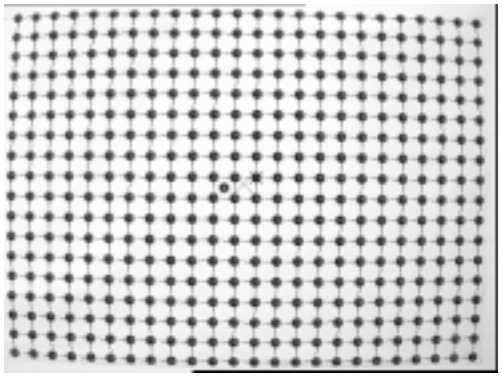
The final result is encoded as parameters for an image rectification program. These parameters are read in before stereoscopic processing, and expanded into rectification tables, which have, for each pixel in the rectified image, the coordinates of the corresponding source pixel in the raw image. During a mapping run, each image digitized by a particular camera is transformed into a rectified image via the table corresponding to that camera.

Figures 2, 3, 4 and 5 on the following two pages show the results of applying **flatfish** to images from 60, 90, 30 and 120 degree field of view cameras, respectively. The **A** figures are the original images, **B** show the regions where the spot operator is positive (small light splotches in the center of the black spots, visible especially in the isolated center spot), and the coordinate grid resulting from linking the spot operator maxima. The **C** and **D** graphs are plots of pixel coordinate radius versus spot coordinate radius for all the spots, **C** for an arbitrarily chosen center marked by a small X on a spot near the center in each **B** image, **D** from the position that minimized the scatter in the graph, marked by a large X. The program assumes the scatter-minimizing center marks the true optical axis. The shape of the curve defined by the scatter diagram is the radial distortion function. The **E** images are the original spot image rectified by the function constructed by **flatfish**, and overlaid with a grid whose spacing should match the spacing of the spots in the rectified image. Figure 5, derived from an image whose field of view was 120 degrees, was rectified to only 100 degrees of view, to eliminate the vignetted corner black areas.

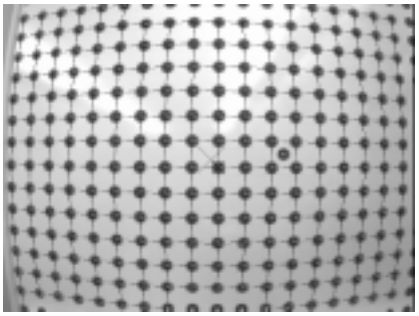


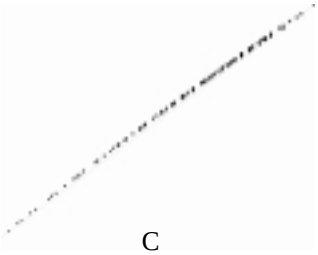
A



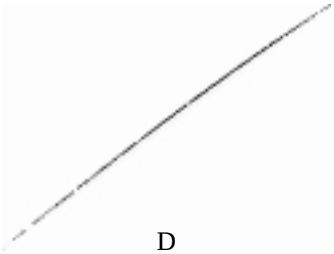


B





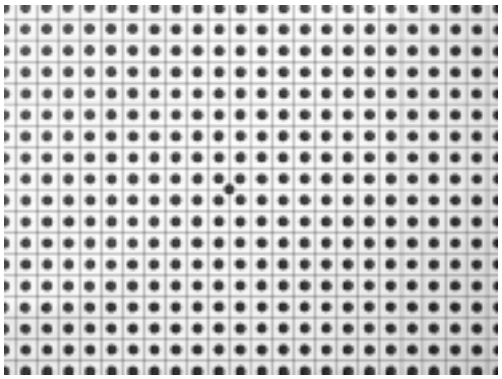
C



D

C

D



E

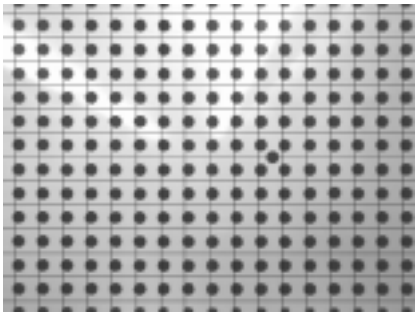
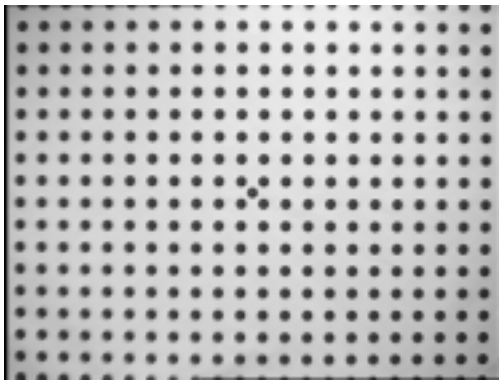
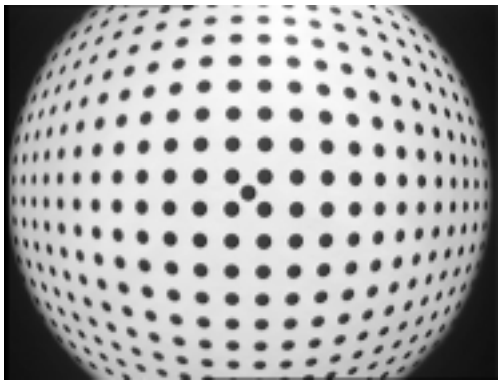


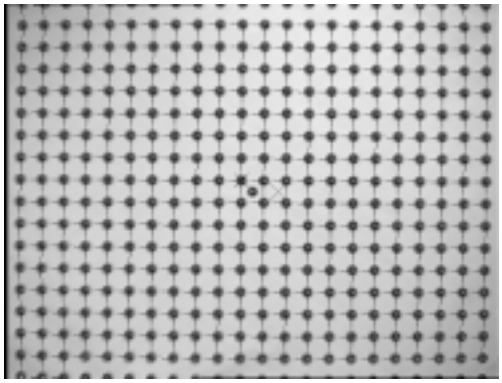
FIGURE 2 *Calibrating 60 degree optics*

FIGURE 3 *Calibrating 90 degree optics*

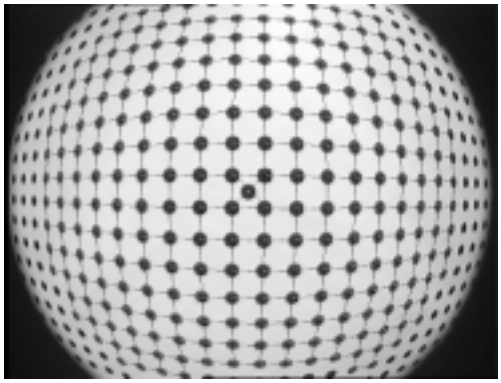


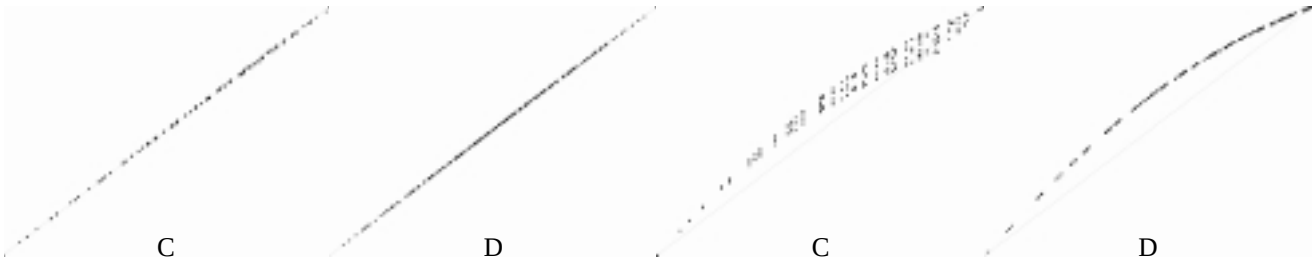
A



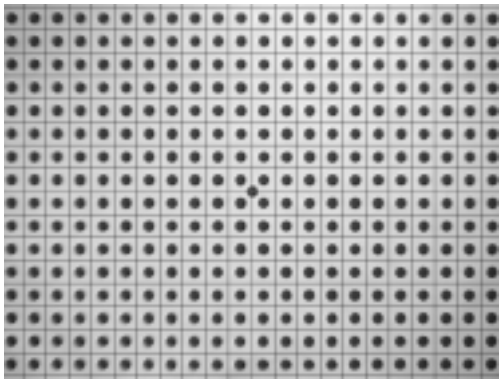


B





C D C D



E

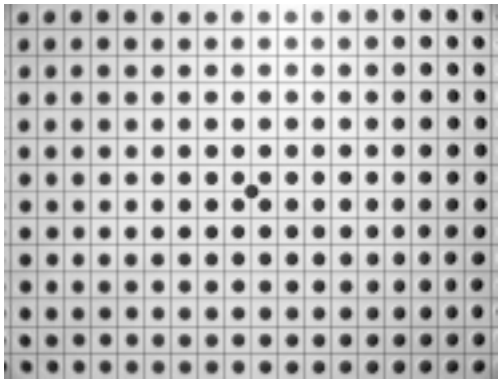


FIGURE 4 *Calibrating 30 degree optics*

FIGURE 5 *Calibrating 120 degree optics*

4 Stereoscopic Mapping Framework

The main image set used in this report was obtained with a pair of Sony XC-999 color cameras, with 6 mm lenses, mounted securely in parallel 15.6 cm apart on an aluminum bar. Twenty stereo pairs of images were collected reasonably carefully by hand, 40 images in total, of one end of an office, in parallel directions from locations on an approximately 50 cm grid on the floor, with the cameras 1.25 meters high, angled 14 degrees down. The scene included portions of walls, two office chairs, two large cabinets and an open door with furniture beyond. One of the cabinets was fully open, and contained a long raincoat and several shelves with small objects. The pictures were originally digitized in color, to a resolution of 768 by 576 pixels, then reduced to grayscale. The resulting images **scan.*.L.pgm** and **scan.*.R.pgm** along with a description of their imaging geometry encoded in a file **run1.nav**, and the two camera rectification files mentioned above **sony60.L.cal** and **sony60.R.cal**, were processed by the stereo mapping program, currently named **crayfish** into a 256 by 256 by 64 cell evidence grid representing a volume 6 meters by 6 meters by 2 meters high.

Crayfish is invoked with the name of a **.nav** file. Its overall flow is as follows:

```
begin
Read .nav file
Read and expand .calib files (indicated in .nav file)
Assign image storage
Initialize 3D evidence grid and sensor models
Loop on images indicated in .nav file
    {
    Read L and R image
    Apply Interest Operator to find distinctive areas in both images
    Loop on windows selected by Interest Operator
        {
       Correlate interest window with possible locations in other image
         Extract probability-weighted peak list from correlation curve
         Loop on correlation peak list
             {
             Calculate 3D location for this peak
             Throw two probability-weighted evidence rays
             }
         }
     }
Write out 3D evidence grid
end
```

A program **wrapfish** takes the 3D evidence grid made by **crayfish** and generates viewable images. As of August 1996, it makes X, Y and Z “slice” images showing all grid planes parallel to the coordinate planes, with very high occupancy probability cells shown in white, very low probabilities shown in black, and intermediate probabilities in shades of gray. It also produces three-dimensional views of the occupied cells of the grid, and a 3D **Open Inventor** file of the occupied cells that can be viewed interactively on Silicon Graphics workstations. The visual quality of the 3D presentations is greatly enhanced by a colorization step that spotlights about a dozen manually selected box-shaped regions covering objects like the floor, walls, cabinets and chairs. In each of these boxes, a region-specific distinctive color is given to all the occupied cells, helping human observers of perspective views distinguish the contents from foreground and background.

The **flayfish** program, derived from **crayfish**, includes additional code to optimize (learn) good parameters shaping the sensor model that defines the evidence rays. It repeats the **crayfish** outer loop and its immediate initialization with different settings of sensor model parameters looking for combinations that maximize a map-quality measure.

Crayfish invokes the separately compiled procedures in files **fisheye** and **volsense**. **Volsense** contains code for manipulating 3D evidence grids. **Fisheye** has procedures for expanding calibration files into rectification tables, and for applying the rectifications to images. It also holds *interest operator* and *correlator* procedures for stereoscopic processing. Following sections describe **crayfish**'s major components.

5 Interest Operator

Stereoscopy in **crayfish** is done by matching small windows in both images of a stereo pair. The position of the matching window in either image defines a 3D ray from the camera, and the relative displacement of the window from one picture to the other defines a 3D position along that ray. The program interprets such matches as evidence for the existence of a visible feature in that 3D location.

Not all locations in an image are suitable for matching. Large areas of blue sky or white wall, for instance, look identical, as do linear regions along simple edges, with no way to distinguish one particular small patch from a nearby one. An *interest operator* is applied to an image to select regions likely to be unambiguously matched in its stereo partner. We invented the concept of the interest operator in 1974, along with the idea of correlation through a coarse-to-fine image pyramid, and used them in the first round of work leading to the present results [Moravec81]. Interest operators and image pyramid correlation are now widely used in the initial registration steps of digital photogrammetry [Hellwich&al94] [Stunz-Knopfle-Roth94]. Mobile robot stereoscopy is much sloppier than photogrammetric stereoscopy because robots, immersed in their scene, encounter visual occlusions, perspective scale and view angle distortions, oblique featureless and specular surfaces, enormous depth ratios and other complications.

After a stereo pair of images is rectified, the program applies an interest operator to the left half of the right camera's image, and to the right half of the left camera's image. Features chosen from those half-images are most likely to be present in the other camera's image, while together still covering the full field of view. A quirk of the approach is that a narrow stripe the width of the camera separation, running to infinity, is seen in both half images, and thus examined doubly.

The interest operator is a variant of one used in our early research. It subdivides the image into a grid of non-overlapping 8x8 windows, and on each computes the sum of squares of differences of pixels adjacent in each of four directions: horizontal, vertical and right and left diagonals. The raw interest measure for each window is the minimum of these four sums, representing the weakest directional variation. If this weakest direction has some contrast, the feature is neither a uniform area nor a simple edge.

Our 1974 interest operator picked only the local maxima of the raw interest measure. The 1996 program applies a high pass filter over the array of interest windows, subtracting the average interest value of the eight surrounding neighbors from the value of each window. Windows are chosen for ranging if they are positive after the high pass operation. The idea in both cases is to cover the image as uniformly as possible, while picking the best possible features in each area. The 1974 program found less than 50 features per stereo set, the 1996 program extracts about 2,500 (figure 6).

FIGURE 6 *Full (omnidirectional) Interest Operator*

The nominal displacement of matching features between the image pairs, from perfectly horizontally mounted and aligned stereo cameras, is purely horizontal - the so-called epipolar constraint. If true, it eliminates the need for vertical correlation search, and allows the use of any features with horizontal variation, for instance simple vertical or diagonal edges. The **flatfish** calibration and rectification process is intended to simulate perfect horizontal geometry, but we have possibly observed that lens iris and focus adjustments can cause image shifts on the order of one pixel, spoiling perfect calibrations. Mechanical shocks might have a similar effect. We are able to detect and correct tiny vertical misalignments after calibration by occasionally permitting a small vertical extent in the horizontal correlation searches, and noting the vertical offset of each best match. A histogram of these offsets from the features from any of the 20 stereo pairs in the data set of this report has a peak at about 3/4 scanline offset. Shifting one image of each pair by one scanline approximately compensates for the misalignment. After the shift, we treat the images as perfectly aligned.

Crayfish exploits the perfect alignment to get more features from the image and to minimize the correlation search. Rather than the full interest operator described above, it uses a version that measures only horizontal variation (figure 7). Unlike the full interest operator, the horizontal-only version strongly registers simple vertical edges, which are very common in indoor scenes.

FIGURE 7 *Horizontal-direction only Interest Operator*

Figures 6 and 7 show interest operators applied to a stereo pair. Each image is the juxtaposition of the left half of a right camera image, and the right half of a left camera image of a stereo pair, a configuration which minimizes out-of-bounds problems in correlation searches. In each 8x8 window, an X indicates a very strong interest measure, a dot indicates a positive but weak measure. Figure 6 shows the full interest operator, which requires variation in all directions, used to determine fine vertical alignments of images. Figure 7 shows the horizontal-only interest operator, used to interpret aligned stereo images

6 Correlation

The *correlator* locates image patches in one image that correspond to interest operator choices in the other image of a stereo pair. As with interest operators, there are two kinds of correlator in **fisheye**, one which searches horizontal and vertical extents, and the other which searches only horizontally. The former is used in occasional image fine alignment steps, the latter is used routinely to interpret stereo scenes.

Omitting complications, described below, the correlator computes a match value between a patch in one image and a corresponding patch around every pixel position in a search area in the other image of a stereo pair. In the existing code, the patches are 7x7 pixel squares. Other sizes also work, and soon we may try circular shapes. The comparison measure is the sum of squares of differences of corresponding pixels, with the window means subtracted:

$$\sum\left(\left(a-\frac{\Sigma a}{n}\right)-\left(b-\frac{\Sigma b}{n}\right)\right)^2 = \sum(a-b)^2-\frac{(\sum a-\sum b)^2}{n}$$

with a and b, implicitly subscripted 1 to n, representing the n pixels in two overlaid windows.

The horizontal-only correlator returns a “match curve” array of correlation values, one for each pixel position in its search. Traditional stereo programs simply select the best match in such curves, but our program processes multiple candidate matches. The full-area correlator searches a rectangle in an image, but returns more information about the horizontal direction. For each horizontal position, it searches the vertical range for the best match, and returns that value in the “match curve” array. A parallel array gets the vertical offset that produced that best match.

The known camera separation, position and orientation for each set of rectified images determines the bounds of the correlation search and the interpretation of the results. Each candidate match is interpreted as a possible 3D surface feature at a particular heading and distance. The evidence ray thrower, described in the next section, adds evidence to 3D grids along narrow cones from the camera positions to the surface feature. With 20 image pairs and 2,500 correlation searches per pair, the amount of evidence accumulated in our 4 million cell grid is substantial, allowing for quite sensitive statistical evaluations of small changes in the program. A very simple such measure is the count of the number of positive cells found in the floor plane of the grid, many produced by weak correlations on the subtle smudges in our uniform gray carpet. About 11,000 floor cells are normally found, but the number drops rapidly with very small deteriorations of the correlator quality. For instance, if the correlation match measure is changed from the means-adjusted version:

$$\sum(a-b)^2-\frac{(\sum a-\sum b)^2}{n}$$

to a simple sum of squares of differences:

$$\sum(a-b)^2$$

the number of floor cells drops to about 4,000. Simply rounding the correlation window means from 16 to 8 bits reduces the floor count from 11,000 to 7,000.

The means adjustment is not uniformly beneficial. It costs computation in the inner loop, and it discards information about the absolute brightness of windows, sometimes finding a best match between a white patch with a black one. In an experiment with a wider than usual correlation search, where only best matches were considered, for 20 high contrast windows drawn randomly from the data set, the simple and adjusted measures both made six errors in twenty correlations, but only three in common. In the cases where the simple measure was correct and the adjusted measure wrong, an absolute brightness difference overrode a spurious similarity in variations. In cases where the adjusted was right and the simple wrong, an overall difference in brightness between the pictures overrode a match in subtle contrast. The simple measure fared worse with low contrast windows, making 12 errors in 20 against the adjusted’s 10 errors. Still, the simple measure was correct in one low contrast case where the adjusted measure was in error.

FIGURE 8 *Correlation of an unanbiguous feature.*

Optical and electronic adjustment differences, and automatic gain changes with changing views, ensure that the images from the two cameras will rarely match each other exactly in brightness. Subtracting out the window means usually improves the correlation by removing such differences. Sometimes it removes too much, and allows a very dark patch to match a very light one. We found a solution to this problem that has the added benefit of speeding up the correlation. A comparison of the means of the ***a*** and ***b*** windows in a correlation can be used as a preliminary discriminant: too different, and a match between the windows can be rejected without doing the expensive $\Sigma(a-b)^2$ calculation. The means can be precomputed for the entire images very efficiently by a sliding sum technique. A preliminary test of the idea showed that about half of the square sums could be eliminated by this approach, while the number of errors in the high contrast example above dropped from 6 to 4 out of the 20.

FIGURE 9 *Correlation with ambiguous matches.*

The program extracts multiple peaks from the remaining correlation curve, each representing a hypothesis of distance to the feature, and translates the match value of each peak into a relative probability using the formula:

$$p_{relative} = e^{\frac{-\sqrt{\frac{\Sigma(a-b)^2 - \frac{(\Sigma a - \Sigma b)^2}{n}}{n}}}{Cscale}}$$

Cscale is an adjustable constant, which will probably be optimized by a future learning process that will also adjust other program parameters. It is in units of grayscale steps, and values of 0.5 to 2.0 seem to give good results. Each $p_{relative}$ is divided by the sum of $p_{relative}$ over all the peaks, to give a normalized probability. The normalized probabilities are used to weight evidence rays thrown for each distance hypothesis. For the examples in this report, the program generated rays for a maximum

of four hypotheses per correlation search. As ***Cscale*** is made much smaller than 1, the probabilities for weaker correlations diminish relative to the strongest, eventually leaving only one peak per correlation.

The above scheme was derived from the following reasoning. Good information about correlation trustworthiness seems to be contained in the array of match values produced by a correlation search. All but (at most) one of the match values are simply from samples of the source window compared against unrelated patches of search image. A histogram of these values looks like a probability distribution with an exponential tail tapering down towards perfect match. A distribution fitted to the histogram, then integrated over matches better than a given one, estimates the probability that the given match is simply a chance coincidence. One minus that probability, is the probability that the match is not random, i.e. correct. Our e^{-match} function for weighting evidence rays approximates this latter probability.

Figures 8 and 9 illustrate results from the horizontal-only correlator. The large pictures show the images being searched. In each, the small square inset, from the other image of the stereo pair, contains the window being searched for: it is the tiny square at the center of the cross of bounding lines. Two vertical lines running the height of the picture mark the horizontal bounds of the search. The narrow horizontal band between these lines is the track of the search window. The graph at the top of the image between the vertical lines is the match value $\sqrt{\frac{\sum (a-b)^2 - \frac{(\Sigma a - \Sigma b)^2}{n}}{n}}$ for each horizontal position, with zero, the best possible match, at the top of the image. Gaps in this curve are locations where the match value was not calculated because the window means, $\frac{\Sigma a}{n}$ and $\frac{\Sigma b}{n}$, differed by more than about 10% of the total brightness range. The bar graph at the right of the image is a histogram of the match values in the curve. The graph at the bottom of the image shows the probability assigned to peaks in the correlation curve by our exponential model. Above a threshold indicated by the dashed line, each peak results in two rays of evidence, one from each camera position, weighted by the probability. Figure 8 shows an unambiguous match where one correlation peak dominates. In figure 9, the black edge of the chair finds close matches in at least four locations, even though half the search area is eliminated because of overly disparate brightness.

7 Evidence Ray Throwing

The heart of **crayfish** is the evidence ray code. In 1990 we noted that efficient two-dimensional evidence grid programs were able to insert hundreds of thirty degree sonar beams per second into 2D grids sized 64 cells by 64 cells using 1 to 10 MIPS of computation, fast enough for real time use on robots of the day. Three-dimensional grids promised a much richer world map, with not only an extra dimension, but higher resolution. Horizontal 2D maps conflate the surroundings at different heights, failing to distinguish projecting handles or overhanging tabletops from basic surfaces, for instance. For this reason, in typical environments, little is gained by choosing 2D grid resolutions better than about 10 cm. The world is consistent in 3D, however, and could usefully be mapped at 1 cm resolution. A high resolution 3D grid, covering an area similar to our several-thousand-cell 2D grids, would contain several million cells. With so many occupancy values to determine, the grid would also be hungry for proportionally more sensor data. This reasoning suggested that 3D grids would be a thousand times as computationally demanding as their 2D counterparts. The author started a 1992 sabbatical year at supercomputer manufacturer Thinking Machines Corporation with this ex-

pectation, intending to gain some early experience with 3D grids using a CM-5 supercomputer. A series of innovations and approximations, implemented with efficient representations and coding in an intense seven-month programming effort, resulted instead in a program **volsense** that was about a hundred times faster than anticipated, sufficiently fast for a conventional computer. This speed made **crayfish** possible in 1996.

The focal point of **volsense** is a procedure to add precomputed evidence functions representing single sensor readings, collectively called a *sensor model*, to 3D map grids. Each element of a sensor model is a spatial evidence pattern representing the occupancy information implied by a possible reading, for instance a particular range from a sonar ping or stereoscopic triangulation. The evidence accumulation operation is a simple integer addition, representing a Bayesian update, with quantities interpreted as probabilities in log-odds form, i.e. $\log\left(\frac{p}{1-p}\right)$.

In original conception, the sensor models for 3D grids would themselves be smaller 3D grids, which would be positioned and rotated relative to the map grid, corresponding to the position and orientation of the physical sensor. A first innovation noted that almost every sensor we considered, including sonar, stereo, laser rangefinders and various proximity and touch sensors, could be approximately represented by evidence patterns symmetric about a view axis. The symmetry allows sensor models to be simple 2D grids, with one dimension radius ***r*** from the axis of view, the other distance ***d*** along it. In use, the ***rd*** planes are swept about the axis into cylinders as they are added to the map grid. A 3D grid map can be seen as a series of ***xy*** planes layered in the ***z*** direction. An ***rd*** cylinder intersects successive ***xy*** planes in a series of ellipses. The effective ***z*** direction can be chosen from among the three grid axes to minimize the eccentricity of these ellipses.

If the ***xy*** coordinates of each plane are shifted to put its origin at the center of its ellipse, the mapping between ***xy*** and ***rd*** coordinates on successive planes becomes very regular. Each particular ***xy*** has identical ***r*** on successive planes, and its ***d*** simply increases by a constant from one plane to the next. This allows an ***xy*** -> ***rd*** addressing ellipse to be precomputed for one plane, and repeatedly reused to fill the cylinder. With all coordinates precombined into single address words in a table representing this ellipse, the inner evidence accumulation loop requires only ten single-cycle operations, mostly integer additions, per cell updated. This approach is at least 10 times faster than straightforwardly combining arbitrarily rotated 3D grids.

Another factor of 4 efficiency came from considering only the cells that change the map, typically a cone radiating from the sensor. A cone has one quarter the volume of its bounding box. At the time ***rd*** sensor models are generated, the sensor model builder also stores a profile of the maximum ***r*** of significant data for each ***d*** value. For each ray cast, the slice precalculation sorts the ***xy*** -> ***rd*** addressing table into increasing ***r*** order. The mapping geometry assures each ***xy*** slice is updated from a V-shaped wedge in the ***rd*** array. For each slice, the evidence accumulation inner loop terminates when the ***r*** value in the addressing table reaches the profile's maximum ***r*** in the current wedge.

A factor of 2.5 speedup was obtained for "free" from optimization level O3 in the 1992 vintage **GnuCC** compiler. This was far better than the optimizations provided by earlier compilers of 2D grid programs. Additional efficiencies occur in the addressing slice precomputation. The ***xy*** -> ***rd*** table is put into ***r*** order by a linear-time bucket sort. The program exploits a four-way skew symmetry in the intersection ellipse, and gets advantages from various coding and incremental computation techniques. For the special case of "thin-rays" never more than one grid cell wide, the program employs much simpler code, about twice as fast as the general "fat-ray" method described above.

In 1992, on a 25 MIPS Sparc-2 workstation, the program was able to throw about 200 wide-beam sonar, or 4000 narrow-beam stereo-vision rays per second through a 128x128x128 world. In 1996, on a 100 MIPS Sparc-20, the speeds are four times as high.

FIGURE 10 *A picture summarizing a stereoscopic sensor model. Slices of evidence rays, destined to be swept into cylinders, are shown as thin dark vertical lines running downwards from the top of the image, ending in a bright spot. The dark lines are the "empty" parts of the rays, the bright spots are the "occupied" range. There is one sensor model for each pixel of possible stereo disparity. Large disparities, representing short distances (about a meter) are on the left. Small disparities, representing large distances, are on the right, truncated at a distance the program calculates will always be beyond the grid boundaries. Rays in the left of the image have a radius of one pixel. Two bands are seen on the right, with beams two pixels and three pixels in radius. The latter will be swept into cylinders about five grid cells in diameter. The gray banding effects are caused by the conical shape of the rays. The radius two and three rays are "contaminated" by zero evidence regions outside their narrow starting points, that are gradually eliminated as the ray expands along its length. Zero evidence is represented by a probability of 1/2, which is rendered as a gray value half way between black "empty" and white "occupied", and so lightens the early parts of these rays.*

8 Constructing Stereo Rays

In 1992 **volsense** contained a procedure for constructing sensor models inspired by our sonar experiences. A sensor model was shaped by 15 parameters, designed to be adjusted by a learning process, controlling evidence cone angle, depth of empty interior, height and thickness of occupied range surface, and how these values changed with distance.

Stereoscopic ranging fit the generic model poorly. Stereo distances and uncertainties are well defined by the geometry of the stereo triangulation. A procedure for constructing stereo-specific sensor models was added to **volsense**. The models it generates contain one evidence pattern for each possible stereo disparity. The beam angle can be varied from that representing a single image pixel, to the width of a correlation window. The examples in this paper were generated with the latter setting. The depth range uncertainty is geometrically derived assuming a single horizontal pixel of correlation uncertainty. The evidence rays have a hand-chosen negative occupancy evidence from the imaging position to the beginning of the range uncertainty region, and a much stronger positive value along the region, diluted by the volume of uncertainty. These parameters and many others will be grist for a future learning process, which will vary them to produce better 3D grid maps. Figure 10 shows a graphic representation of a stereoscopic sensor model.

Crayfish throws two rays for each correlation peak, one from each camera position, intersecting at their calculated range. Multiple hypotheses derived from a correlation curve are translated into multiple superimposed pairs of rays. The evidence in each pair is weighted by its match probability. This is accomplished efficiently by selecting one of several precomputed sensor models with variously diminished evidence values. We typically configure the program to generate 8 or 16 complete sensor models, coarsely representing probabilities 0 to 1.

9 Program Speed

On a 100 mips Sparc 20, a typical run of **crayfish** reports the following average timings per stereo pair processed for the program's major steps. Each pair generates about 2,500 correlations, each resulting in 2 to 8 evidence rays, depending on ***cscale***, and limited by another program parameter.

Rectification of two images:	0.2 seconds
Interest Operator:	0.1 seconds
Correlator:	0.9 seconds
Ray Throw:	0.7 to 2.5 seconds, depending on ***cscale***

10 The Office Scan

Crayfish was developed with a data set collected using a pair of Sony XC-999 color cameras, with 6 mm lenses, mounted securely in parallel 15.6 cm apart on a tripod-mounted aluminum bar. Twenty stereo pairs of images were collected reasonably carefully by hand of one end of an office, in parallel directions, from locations on an approximately 50 cm grid on the floor, with the cameras 1.25 meters high, angled 14 degrees down. Figure 11 is a view from the back of the data set, and encompasses most of the scene. Figure 12 shows the placement of the cameras in the room.

FIGURE 11 *An overview of the room first imaged by **crayfish**. Twenty pairs of images were processed, from the view position of this illustration and forward, stopping at the location of the chairs.*

Wrapfish produced figures 13-16 from the 3D evidence grid output by a **crayfish** run on the room images. Figures 13-15 show slices through the evidence grid. Black means "empty", white signals "occupied", while middle gray indicates no evidence. Each image consists of a number of black-bordered frames representing successive slices through scene, in the sequence left to right and top to bottom. Tick marks around the frames represent 1 meter distances in the scene.

Figures 13 and 14 are horizontal slices in the same orientation as figure 12. Each frame maps 3 1/8 vertical centimeters. Figure 13 covers 0 to 19 cm high. The first three slices in Figure 13 are covered with a dense mat of white spots representing the floor. The three or four prominent parallel lines on the left come from correlations on grooves in the tiled floor outside the office door. There are no corresponding perpendicular traces because the interest operator rejects features with only horizontal variation as unsuitable for matching by horizontally separated cameras. The two five-pointed stars seen in the latter three frames are the wheeled bases of the two chairs. Other prominent features are the open cabinet door, the base of a wall at the bottom of the images and the black swath of "emptiness" radiating out of the office door. Small but strong features include the office door frame, the

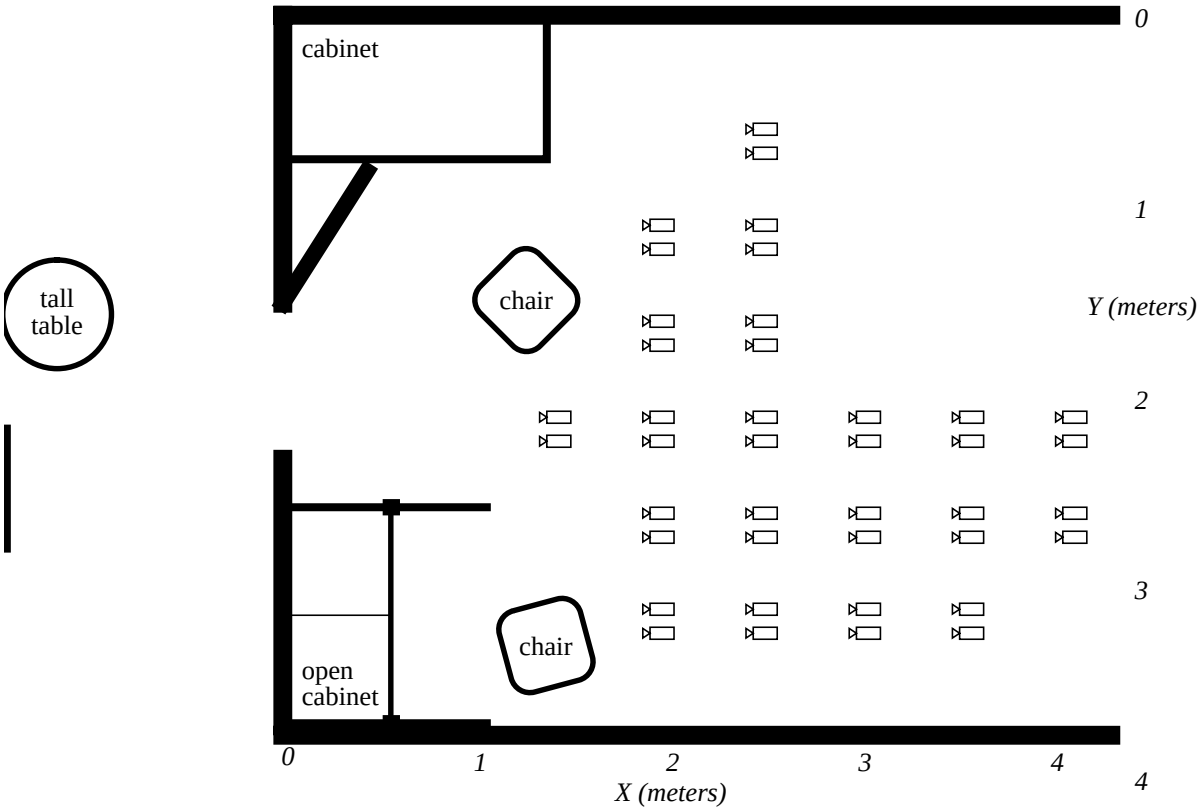


FIGURE 12 *A schematic overview of the room mapped in the first runs of **crayfish**. Camera positions are marked by camera-pair icons, looking to the left. The image in figure 11 was taken by the camera just left and below the Y = 2 meter mark.*

rumpled raincoat in the open cabinet, and the base of the tall table. Figure 14 contains slices from 38 to 56 cm above the floor, and shows the seats and some of the backs of the chairs.

Figure 15 shows vertical slices 2.3 cm thick, parallel to the wall containing the door, from 15 cm outside to 35 cm inside the office. Prominent features are the door frame, the portion of wall with the light switch, the raincoat, and cabinet shelves with several small items. The raincoat's many wrinkles and shadows made it an especially effective subject for the interest operator and correlator.

Figure 16 is a 2D projection of the 90,000 "occupied" cells of the 3D grid. Much information is left out in this image, including the relative strengths of the occupied cells, all the contents of the 1.5 million "empty" and 2.5 million "unknown" cells, as well a an entire dimension of separation. Some depth cues are restored by "spotlighting" 3D volumes in distinctive colors. All the occupied cells in about a dozen box-shaped volumes have been given box-specific colors. One box colors the floor layers cyan. Each of the chairs is enclosed in a box of black color, the door frame is in a magenta box, the coat is encased in red, and so on. Occupied cells outside the major colored areas, probably the result of correlation errors, are colored yellow, and give the image a noisy background. The result lacks the precision and details of the slice images, but does a much better job of presenting the totality of the grid.

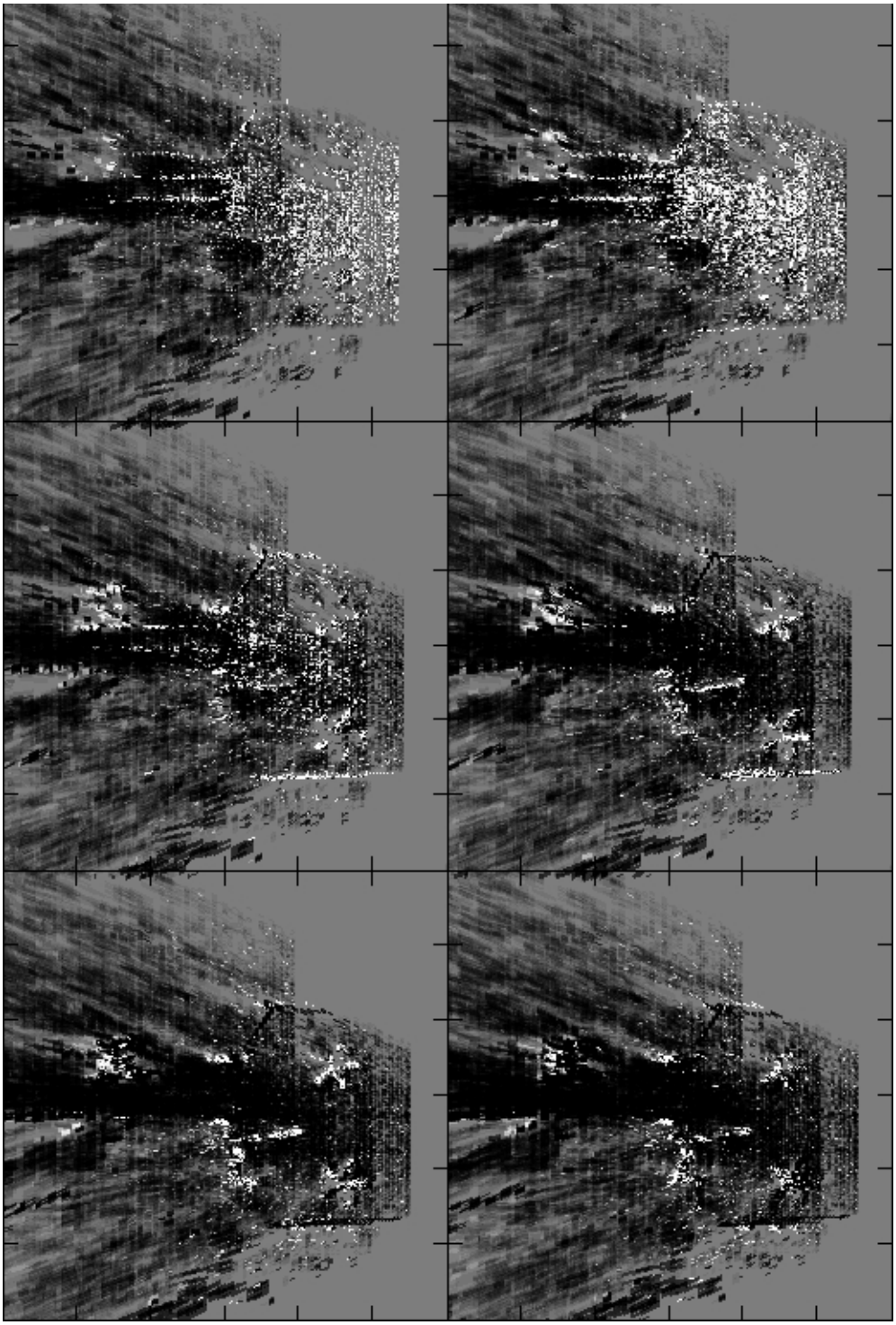


FIGURE 13 *Horizontal slices representing the first 19 cm above the office floor, oriented as in figure 12. The carpet, grooves in tiles, the base of the chairs, a cabinet door, a long wall, the office door frame, a raincoat, the base of the tall table and other features are discernible.*

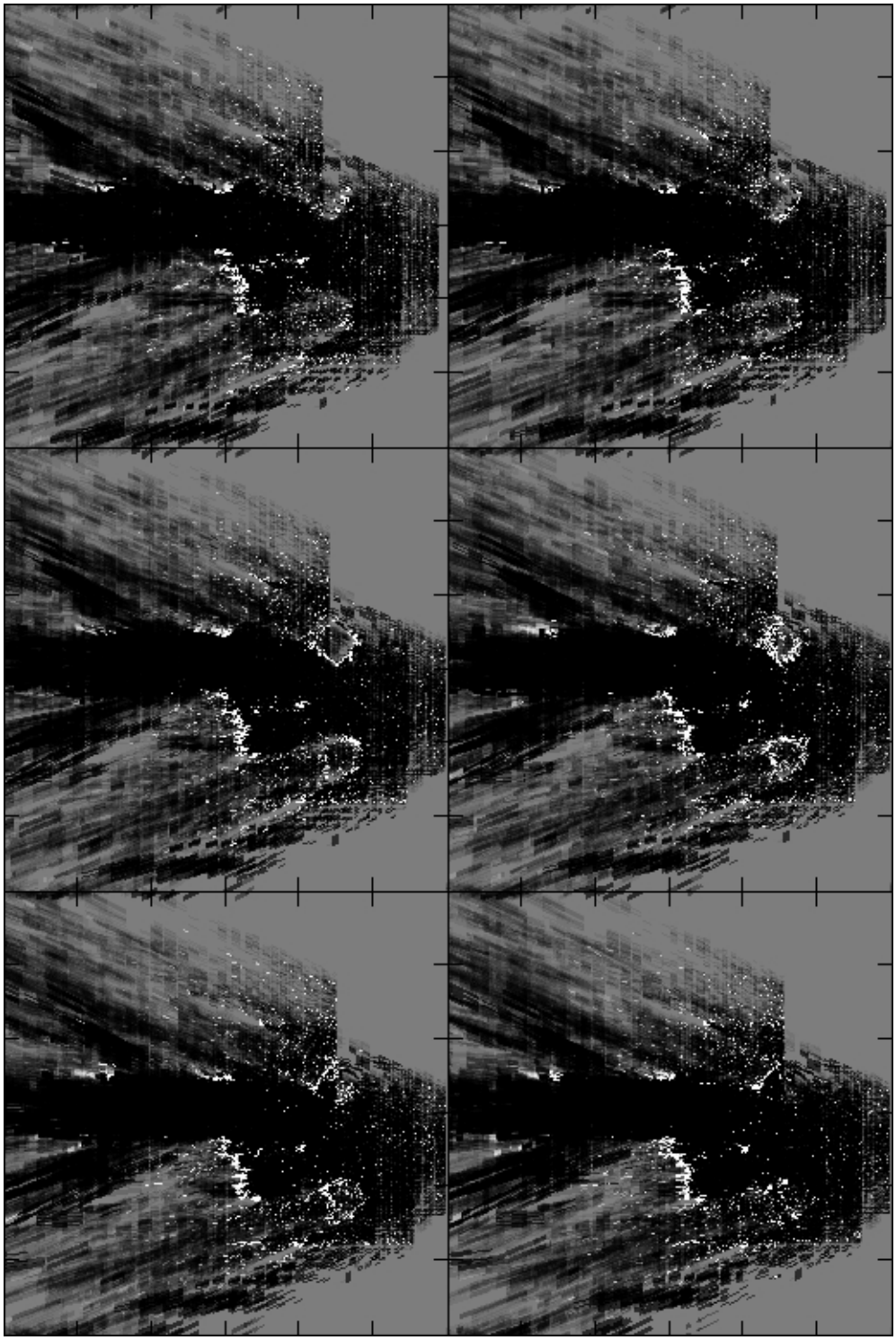


FIGURE 14 *Horizontal slices 38 to 56 cm above the floor. The seats, backs and arm rests of the chairs make an appearance.*

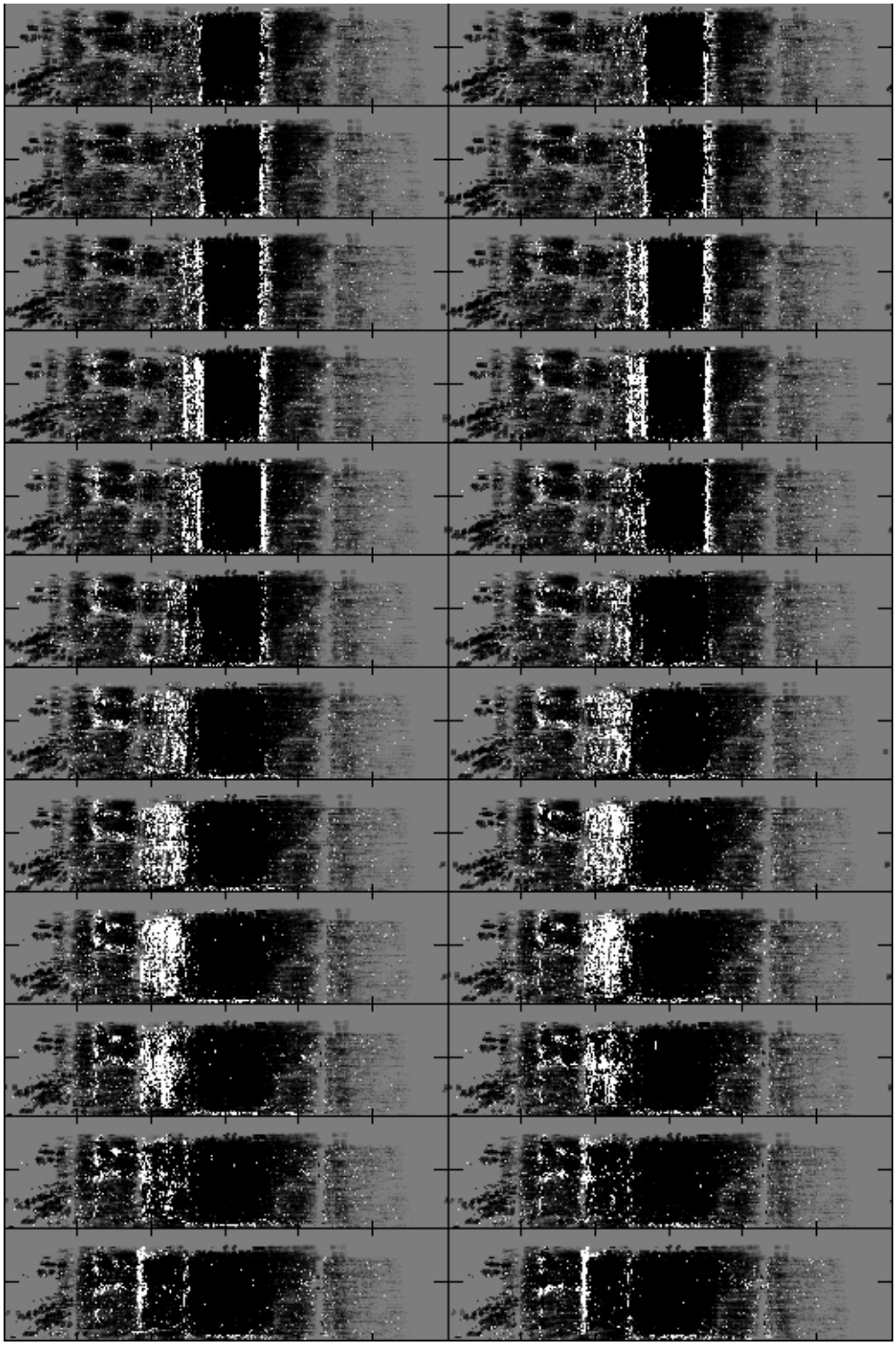


FIGURE 15 *Vertical slices moving towards the camera positions, from just outside the office door to just inside. The door frame, the raincoat and the cabinet shelves and their contents are visible.*

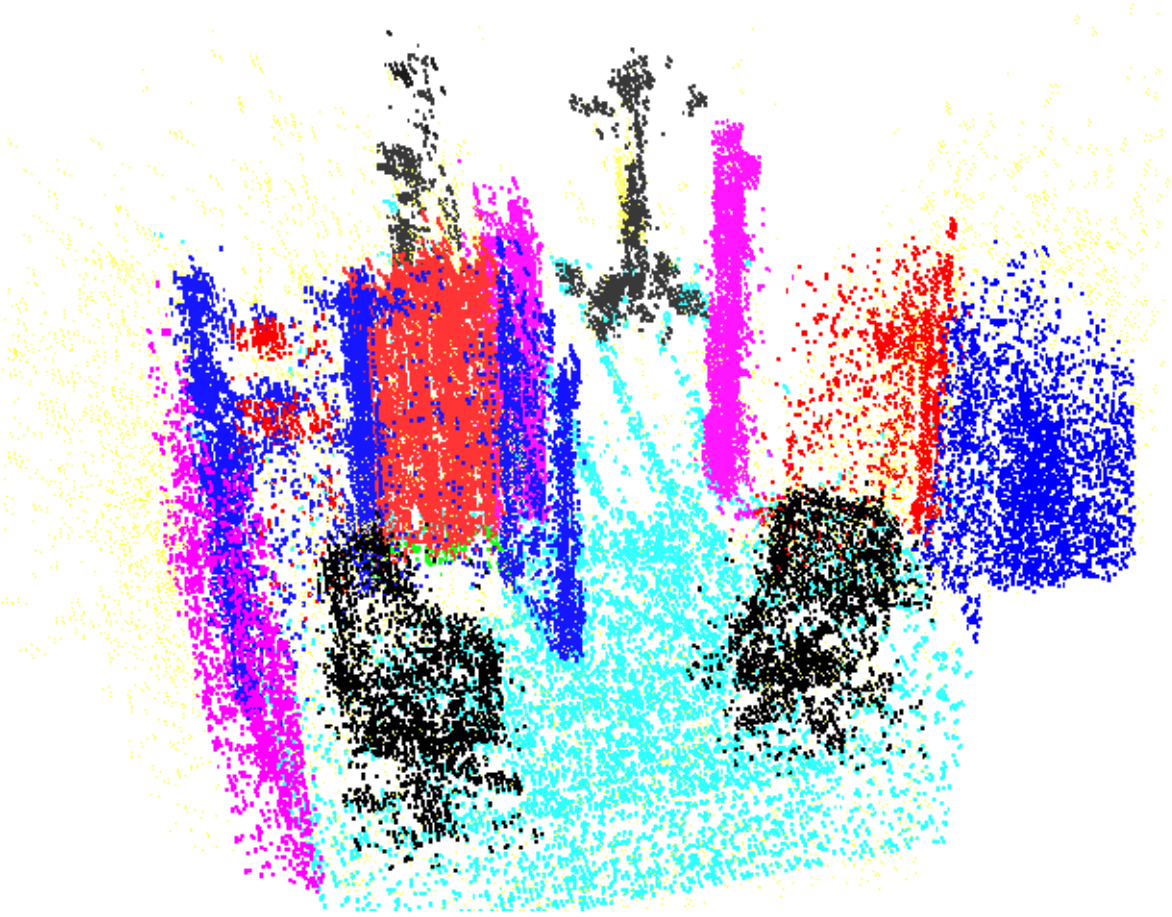


FIGURE 16 *A 2D projection of the 90,000 occupied cells in the 4 million cell 3D evidence grid of the office. About a dozen box-shaped regions have been "spotlighted" in color for clarity - a clarity unfortunately reduced in monochrome reproductions.*

Wrapfish produces an even more dramatic display, in an **Open Inventor** 3D data file that can be viewed on Silicon Graphics workstations. A scene similar to Figure 16 results, that can be interactively rotated in 3D to give a much better sense of the volume relationships.

11 Work in progress: Learning 3D Sensor Models

Crayfish produced encouraging results in its first runs, with correlation and sensor models hand-chosen with modest thought and no experience. It can surely be improved, for instance in background noise level and object surface definition, with better sensor models. A learning process, optimizing a handful of parameters, made a dramatic improvement in a sonar/2D grid experiment [Moravec-Blackwell93], and we hope for similar gains in the new 3D context.

Automatic learning requires a criterion to optimize. In the sonar/2D example we were able to hand-construct an "ideal map", or ground-truth model, against which reconstructed grids were compared. The 2D ideal was simple, containing only a few walls and doors at low resolution. The same approach is impractical in our high resolution 3D scenes, which contain tens of thousands of tiny details at the scale of the grid. Ground truth could perhaps have been derived from a high quality scanning laser rangefinder, but none was available while our scenes existed, nor is it now.

Several weeks of experimentation failed to produce a good learning criterion among measures that compute various kinds of local solidity in the grid data. Some drove the grid to zero density, others filled it with noise. Some had parameters that could be adjusted between these extremes, or to achieve some specified average density, but none were convincingly measures of the correct answer. We considered hand-massaging the best 3D grids encountered to stand in for ground truth, but a more broadly useful approach has suggested itself.

There is a kind of ground truth in the original images of the scene, albeit in 2D projection and with surface color. The grid, on the other hand, is 3D, and has no color. It is easy to project the grid's 3D cells into 2D pictures (figure 17), recreating the geometry of the original images, but without color the grid projections convey little information. But original colors for the grid can also be obtained from the images!

We plan to evaluate trial 3D grid maps by dividing the images from which they were derived into a "coloring" set and an "evaluation" set. In the 20-pair office data, images from more forward positions are probably best for coloring, and those looking from the back for evaluation.

We will scan the 3D grid from far to near and make a list of positive ("occupied") cells in scan order. Typically there are a manageable 90,000 occupied cells out of the 4 million total cells. Each element of this list will get the grid coordinates of the cell, and also an, initially blank, "cell color".

The cells in the list will be rendered into the viewing geometry of each of the "coloring" images, producing corresponding synthetic images, with each pixel indicating the identity of the closest occupied cell visible at that position, or nothing, if no cell projects there. The program will scan the synthetic images, and, at each non-empty pixel, average the color of the corresponding location in its "coloring" image into the "cell color" variable of the indicated cell. When done, each cell in the "occupied" list will have an associated color, averaged over the images in which its was visible.

Then the program will project the newly colorized grid into the geometry of the "evaluation" images, and compute similarity. The similarity test is like the colorization, but instead of averaging the original image color with the cell color, the two are compared.

The evaluation is very similar to a human test of a reconstructed grid: it asks the question, "does it look right?". The approach uses information, and measures interesting qualities, in the source images not used in constructing the original grid. For the initial implementation, we will probably use monochrome images, averaged down to half size, to do the (grayscale) coloring and evaluation. Color images also exist, and we could use those later. A very interesting side benefit of the colorization process is a grid with the original scene colors, which should be exciting to view.

12 Things to Twiddle

The 3D sensor-grid's many unexplored directions will surely provide years of amusement. A working learning process and a sensitive map-quality criterion will provide the opportunity to optimize all parts of the process.

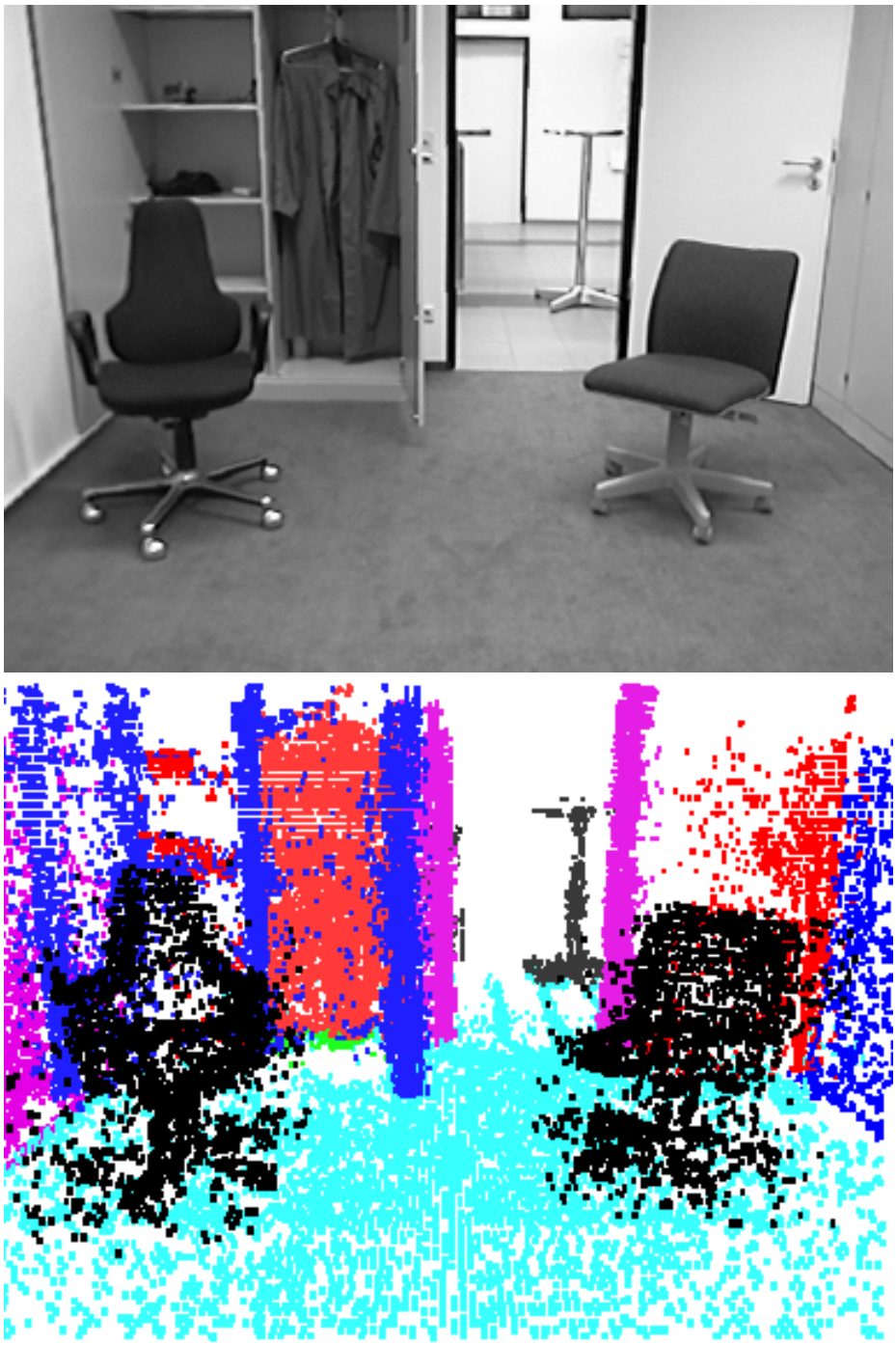


FIGURE 17 *An original image and a synthetic image with the same imaging geometry. The synthetic scene colors were chosen by hand. When the learning program is complete, the colors will be natural, derived from other images of the scene.*

Many variations in the basic stereoopsis suggest themselves. Would it be better to preprocess the image for features like edges? Should the correlation window be round to maximize the number of pixels for statistical significance, while minimizing the distance from the center? Should the windows be taller than wide, because a horizontal change in viewpoint causes more horizontal than vertical image variation? Should they vary in size? Would pre-adjusting the brightness and contrast of the images on the basis of the strong correlations improve the weak correlations? How would this interact with the brightness-difference cutoff threshold in the correlation search? Could the minimum distance considered in correlation searches for weak features be modified from image to image, or even region to region, on the basis of strong correlations from the same area? The search costs and error rates are minimized if the search is narrowed as much as possible. How much improvement can be achieved by using other interest operators? Should they return more or fewer points? What about different comparison measures? How should evidence rays be weighted as a function of correlation quality? And many more.

Even more fundamental to the grid function is the sensor model, i.e. the shapes and weights of the evidence rays. The existing program uses a very simple, ad hoc sensor model for stereo evidence. An evidence ray, which is often a line, but can become a cone a few cells in diameter, consists of a hand-picked negative value from the camera to the region of stereo range uncertainty, followed by a positive value covering the length of the uncertainty. The positive value is another hand-chosen constant divided by the size of the positive volume. The weights are scaled by the correlation probability. The evidence grid approach derives trustworthy conclusions from very noisy data by accumulating large amounts of it for each spatial location. The accumulation process is compromised if the individual evidences have systematic biases. A good sensor model is a bias-free representation of the average true information from each kind of measurement. A learning process that adjusts angles, weights, sizes, shapes, dependence on distance and correlation and many other subtleties should converge on such a sensor model. Our simple hand-picked model is unlikely to be very close.

For some purposes, including producing synthetic images, it is necessary to classify evidence values into empty, occupied and unknown. The threshold values for these classifications could be optimized during learning.

Some important improvements are beyond the reach of simple learning. This report's data set was collected by hand, with tripod mounted cameras. The camera positions cameras varied at least a fraction of a centimeter from nominal, and the pan and tilt angles may have varied by more than one degree. The latter errors, especially, mean measurements from different views were not perfectly registered. Images from moving robots are likely to have even larger position and orientation uncertainties. We and others [Yamauchi96], [Schultz-Adams-Grefenstette96] have had good success in past registering sonar/2D grids of the same area made from viewpoints, whose relationship is known only approximately, by searching for best map match over a range of relative positions and orientations. The large number of cells in the grids allow registration accurate to a fraction of a cell, by interpolation of the discrete matches. This works even when one of the grids is built with only a few dozen sonar readings from a single robot position. The statistically stabilizing multitude of cells in the 3D grids assures us that the approach will work even better there. The only question is speed: our grid has four million cells, and a search over large ranges of 6 degrees of freedom could easily take trillions of computations! Fortunately there are many strategies for enormously reducing the cost. For a mobile robot on a flat floor, only 3 degrees of freedom vary very much, so the search extent in the other three can be very small. Even the unconstrained position and angle searches can be kept small by using dead-reckoning information. The search can be reduced almost to its logarithm, by a coarse-to-fine strategy, where approximate answers are computed first at the small end of a pyramid of shrunken grids. Coarse-to-fine has worked very well for us in past, in unconstrained stereoscopic correlation and in matching 2D grids. Well-chosen samples could substitute for the whole: for instance, the un-

known areas of a grid could be skipped, or the occupied cells of one grid could be mapped to the entirety of the other. A branch and bound strategy might cut short some of the comparisons, when their accumulating partial errors exceed the best previous comparison, a method that works best if the most likely registrations are searched first.

We hope to implement an image-pair to scene registration step soon after the learning code is working. It is likely to tighten up the office scene, and will prepare the way for stereo/3D grid use on real robots.

13 Hardware Helpers

Quirks in our office scene reconstruction suggest some easy hardware alterations likely to significantly improve performance. The dense coverage the correlator obtains on the slightly textured carpet, and more heavily textured raincoat, compared to its sparseness on plain surfaces like doors and walls, suggests that coverage would be greatly increased by the artificial addition of texture across the scene. This could be accomplished simply by mounting a randomly textured floodlight with the cameras, perhaps infrared to minimize effects on humans.

More speculatively, wide angle sonar could provide the range to the nearest feature in the scene, to narrow the correlation search to that range and beyond, boosting speed and accuracy. A preliminary stereoscopic preprocessing of strong features might give similar information without extra hardware.

Precise image rectification allows the correlation search to be exclusively in the direction of the axis horizontal joining the two cameras. The points to be searched for, selected by an interest operator, need thus have only horizontal variations. As a result, the program picks out a dense coverage of features along vertical wall and furniture boundaries, but entirely misses the horizontal boundaries. This problematic blindness could be eliminated by adding a third camera to the stereo pair, vertically displaced from the original two. The trio could be used in pairs, with points selected by interest operators specific to each pair's separation direction.

The 60 degree field of view cameras, though placed on most of the open floor area of the office, saw only a tiny portion of the left wall and none of the right, and viewed other elements of the scene from a very narrow range of angles. The grid might be much enhanced if objects were viewed from many directions. Wider angle lenses would help, as would panning the cameras from side to side during operation, or having several camera pairs looking in different directions. The latter option is probably best, as manufacturers begin to market high quality, inexpensive single-chip cameras with integrated optics, for applications like hand-held videoconferencing. Simply processing more images would also help, which will surely happen when the cameras are used on a mobile robot that can acquire new sets of images perhaps every second.

Before the fast ray-throwing code was written, speed seemed the bottleneck for 3D grids. Now **crayfish** is quite speedy, but encounters memory limits. Experiments with existing data and small grid extents (see the appendix) show that scene details improve as grid cell size is reduced, even to below 1/2 centimeter. The speed cost for increased resolution is modest. For a wide range resolutions, most stereo rays remain one cell in width, and the time to throw rays grows only slightly more than linearly with grid resolution. The image processing steps are unaffected, so total runtime grows perhaps 50 percent per resolution doubling. Memory usage, on the other hand, grows dramatically. Our 256x256x64 grid has four million two-byte cells, consuming about half the program's 16 megabytes. Doubling the resolution balloons the grid to 64 megabytes. Lowering the cell size to 1/2 centimeter would require a gigabyte of grid. Memory costs are dropping, and sizes increasing with a time constant of about 2 years, so even such numbers should soon be within reach. It's comforting to know

that the existing approach will produce even better results in future, simply through hardware progress.

14 Future Extensions and Applications

Crayfish builds 3D maps from a set of image files, a first step to myriad goals. Its descendents will process real-time images from moving robots. The robots will use their 3D understanding of the surroundings to navigate confidently, to locate and interact with objects, eventually to plan and execute complex tasks involving locomotion and manipulation. To do all that, they will need application programs to extract specific answers from the 3D grids. Here are few answer extractors on our "to do" list:

14.1 Obstacle avoidance

Our 3D grids contain over 100 times as much information about robot surroundings as the 2D maps or sparse 3D models derived by previous mobile robot programs. Since robots can be made to maneuver reasonably well with the sparser information, we have very high expectations for 3D-grid based local navigation. In particular, corners, horizontal projections and inconveniently placed small obstacles, that can be missed or misinterpreted by the simpler approaches, sometimes with serious consequences, appear clearly and unambiguously, in shape and position, in a 3D grid, ready to be noticed by a path-planning program. Finding paths in a 3D grids need not be extremely computationally costly. Since the grid specifically marks empty volumes, the frequent case of an unimpeded path from source to local destination can be quickly detected, by sweeping a 3D grid volume in the shape of the robot along a straight path in the 3D map. The collision probability at each position is found by multiplying the products of occupancy probabilities of corresponding map and robot cells. The outer product becomes a more tractable a sum by taking the logarithm of the inner products. This computation can be made very cheap by a coarse-to-fine strategy, which refines the resolution only for those few cells where coarse comparison detects a possible conflict. A full path planner could be constructed by embedding such a measure in an A* search. By doing its work in 3D, the planner could produce paths that scoot the robot under overhangs and squeeze its shape through commensurate gaps.

14.2 Navigation

Nearly every indoor mobile robot task requires the robot to return to previous locations. Existing commercial delivery and cleaning robots do this with the help of buried wires, wall or floor mounted beacons or markers, or painstaking handmade site-specific maps of walls and openings, whose frequent encounter corrects odometric dead-reckoning. Experimental robots that navigate with autonomously made 2D maps (e.g. [Koenig-Simmons96], [Schultz-Adams-Grefenstette96], [Yamauchi96]), are not quite reliable enough for commercial use, which demands months of operation without navigational failure. Using million-cell 3D maps, with 100,000 surface cells, instead of thousand-cell 2D maps, containing at most a few hundred surface points, should decrease the failure probabilities enormously. Suppose we want a robot to autonomously repeat routes shown it on a guided tour. During the tour, the robot would build and store a sequence of 3D map "snapshots" of the journey. When on its own, it would match these snapshots against its fresh 3D. The relative position and orientation of best match would tell the robot its location relative to its tour route--to a fraction of a cell, if it interpolates. The matching can be done efficiently using the same coarse-to-fine and selective sampling techniques suggested in the last section for precisely registering new data to old in a developing map. The 3D grids contain so huge a mass of detail, at large and small scales, that the proba-

bility of a good match between two maps of the same area, at other than the correct placement, should be astronomically small. By contrast, with simple line maps or blurry 2D grid maps, the probability of a false match, and consequent serious navigational error, is significant. Enormous reduction in navigational and shape confusion probabilities is a major 3D grid promise.

14.3 Object recognition

Volumetric matching, as suggested for navigation, might also be used at smaller scales, and possibly higher resolutions, to detect objects by shape. Grid prototypes of objects can be matched to map areas in the same way maps are matched to each other. At a given resolution, the number of cells in a small object, and thus the reliability of the match, will be lower. Reliability can be improved by increasing the grid resolution, in an approach we may call “fine-to-finer”. Stereoscopic data collected for navigation can be used again to map small volumes at sub-centimeter resolution, as shown by the chair example in the appendix. There are complications. Office chairs, for instance, have parts that move with respect to one another. The back, seat and base of the chair could be sought volumetrically separately, and their match values combined in a formula, with measures of the relative geometric positions, in a hybrid “chair operator” that could be applied to likely positions in the scene. Scene colors could also weigh in the formula. Experience may suggest a bag of such recognition techniques, which can perhaps be orchestrated by an object description language, to conveniently construct operators for even changeable objects like chameleons. A similar approach should work for general navigation. Experience with the language may then suggest how to automate the object and route description processes.

14.4 Scene Motion

The data sets for **crayfish** were accumulated over several hours, with no regard to intervening changes in the scene. If there had been changes, the “before” and “after” states would have been averaged together. At each location, the program accumulates a weight of evidence of occupancy. If a given location has been identified as occupied many separate times, this weight can become very large. If a physical scene rearrangement subsequently empties the location, it will take very many additional observations to erase the weight in the grid, and replace it with a negative one. We could deal with overly persistent memories by a deliberate “forgetting” process. As each update is made, affected cells would have a small proportion of their old evidence trimmed, thus exponentially decaying evidence over the number of sensings of an area, and giving new measurements relatively higher weight. An object moving through a repeatedly sensed area would then show up as a fading streak in the grid, just like the slow-phosphor trail of a moving object on an old-style radar screen. Decay over sensings would leave recently unsensed map areas unchanged. There are reasons to decay those also, but probably at a different rate. If, in its wanderings, the robot steadily accumulates navigational errors, the registration of very old data with new will become uncertain. This could be accounted for by spatially blurring the old data - we have demonstrated this in 2D grids [Elfes87]. All these techniques will become more effective as increasing computer speed allows us to do more frequent sensings. With 500 MIPS, it should be possible to process stereo images about once per second. When computer power and memory several times greater becomes available, it will become possible to take the grid idea into the fourth dimension, with 4D grids representing the local spacetime. Doing so would replace motion blurs with a stack of instantaneous frames of the scene, from which detailed moving object dynamics could be derived. For the next several years, however, we will have our hands full with 3D grids.

15 Conclusion

Since 1984, two-dimensional grids have proven effective in turning mobile robot sense readings into reliable maps, even when individual readings are very ambiguous or noisy. The grid approach catalogs the surroundings by location, a division independent of individual measurements or their errors, allowing arbitrarily many individual readings to be accumulated before any conclusions need be drawn. It tames sensor noise with large-number statistics. Most competing approaches are much less tolerant of noise, because they must decide on interpretations, for instance about surfaces, with far less data. The grid approach threatened to require more memory and computation than prior methods, but its simplicity and regularity allow very efficient implementations. Many experimental mobile robots today are controlled by real-time 2D grid programs.

Two dimensional grids offer, at best, a blurry view of a slice of the world. We have considered them a prelude to 3D grids, which could represent the complete surroundings in precise detail, and constitute the basis of a high-quality robotic sense of space. Their apparent 1,000 times higher computational cost delayed the effort to try 3D until 1992, when a supercomputer became available. Work towards a supercomputer implementation produced instead a surprisingly fast uniprocessor program for the key 3D evidence-ray throwing step. A complete prototype 3D system, described in this report, was finally completed in 1996.

Though it uses unoptimized ad-hoc sensor models, the 3D grid program clearly demonstrates the effectiveness of the approach, and that 500 MIPS would suffice for a real-time version. From stereoscopic range data with 20 to 50 percent errors, it constructs maps that, imaged, look like doll-house replicas of the sensed area. The existing program is good enough to feed a highly reliable robot navigator, but we anticipate increasingly better results, as nearly every portion is scrutinized and optimized.

16 Appendix

The nice results from our initial office image set encouraged us to collect a second, larger, example. At the beginning of August 1996, Christopher Priem collected 59 stereo pairs of images of a busy 8 meter by 8 meter laboratory, with the stereo cameras at a height of 75 cm, aimed horizontally. In part of the data set, the cameras looked from the back of the laboratory towards the front. In a second portion, they looked in a 45 degree diagonal direction, in a third set they looked from the side, and three stereo pairs looked backward from the front. Images from the diagonal and backward-looking sets are shown in Figure 18.

Though the physical area mapped in the second data set was larger that the 6 meter room of the first set, memory limitations prevented us from using a larger grid. The resolution of the 256x256x64 laboratory grid cells is thus only 3 1/8 cm in all three axes. Figures 19 and 20 show horizontal slices through the resulting grid, at floor and chair-seat heights, with the same encoding as figures 13 and 14. Figure 21 contains horizontal slices, as seen from the front of the room looking back, through a chair, two boxes and cabinets. Figure 22 is a colorized projection of the approximately 90,000 occupied cells of the entire grid.

The reduced grid resolution results in a lower quality reconstruction. Memory limitations prevented us from increasing the entire grid size, be we did an experiment with higher resolution, by using all the data to map only a small portion of the lab volume, namely the cubic meter enclosing the chair at the middle front of the room. A 128x128x128 grid for this volume has cell resolution of 0.78 cm. Figure 23 is a projection of the high-res occupied cells. We hope, in future, to map larger areas at such resolution.

FIGURE 18 *Views of the laboratory imaged in the second **crayfish** test. Fifty nine pairs of images were processed, looking forward, left-forward diagonally (as in the top image), from the left side, and three pairs facing backward (bottom image).*

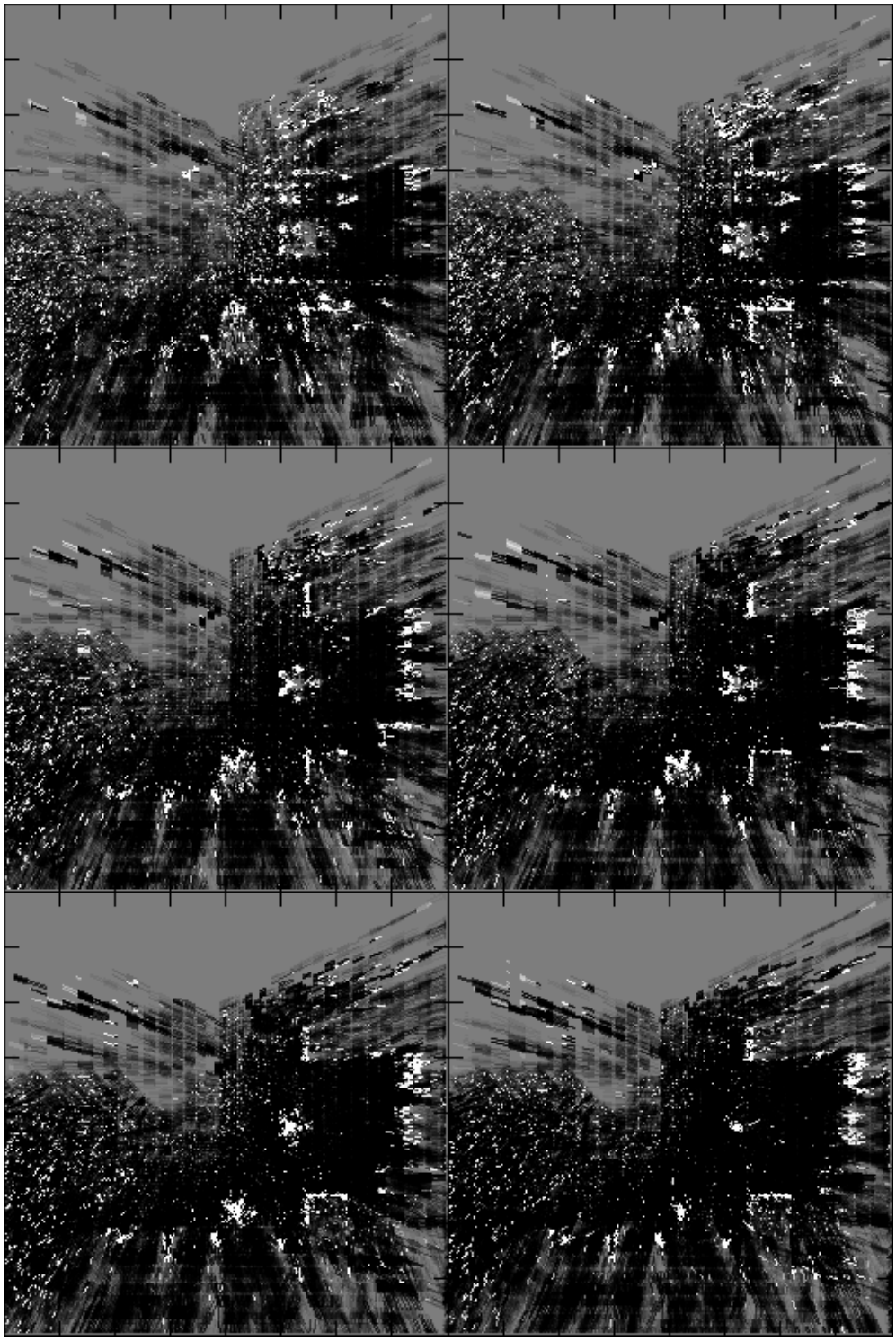


FIGURE 19 *Horizontal slices representing the first 19 cm above the laboratory floor. The front of the laboratory is to the right, the door is at the bottom.*

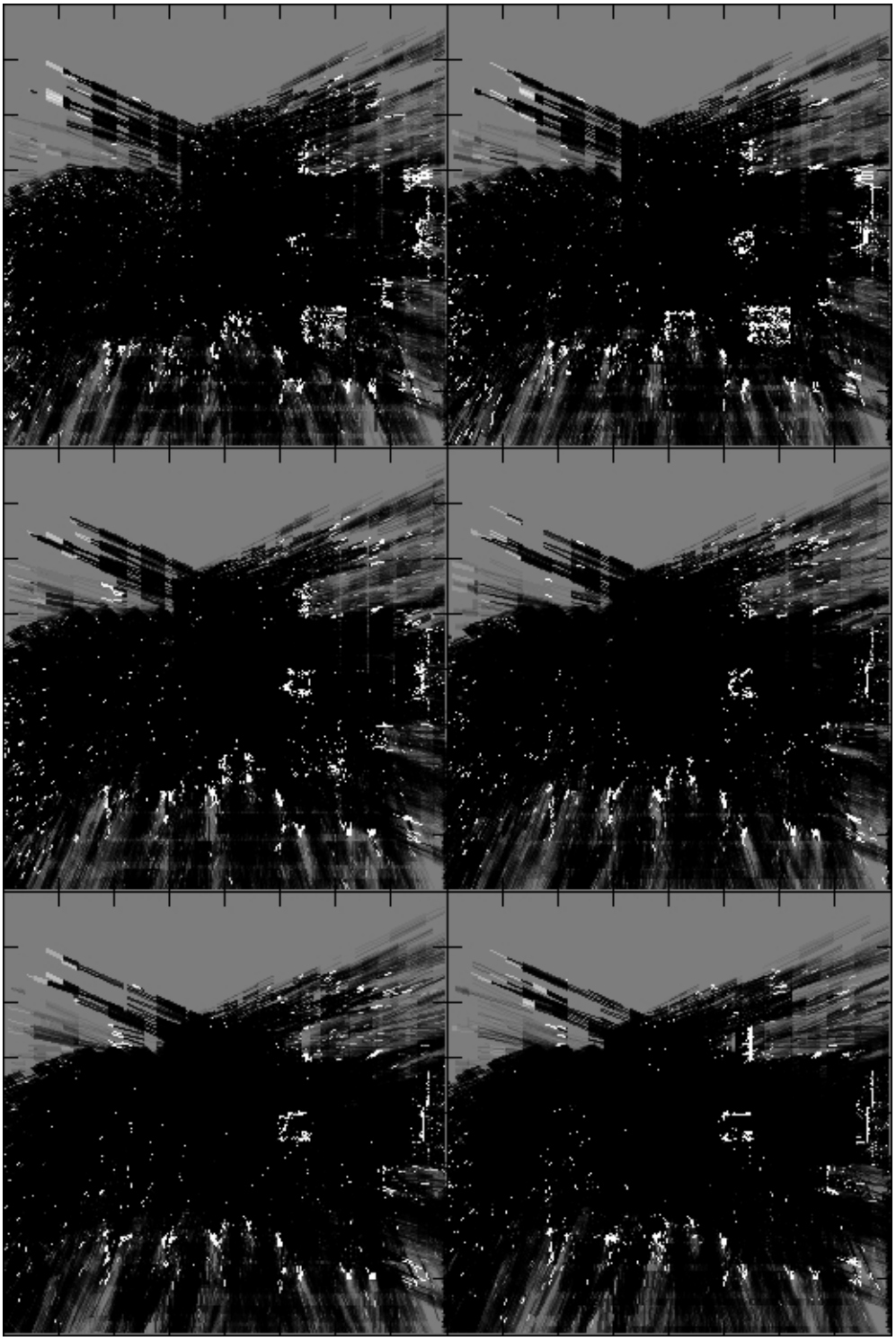


FIGURE 20 *Horizontal slices 38 to 56 cm above the laboratory floor.*

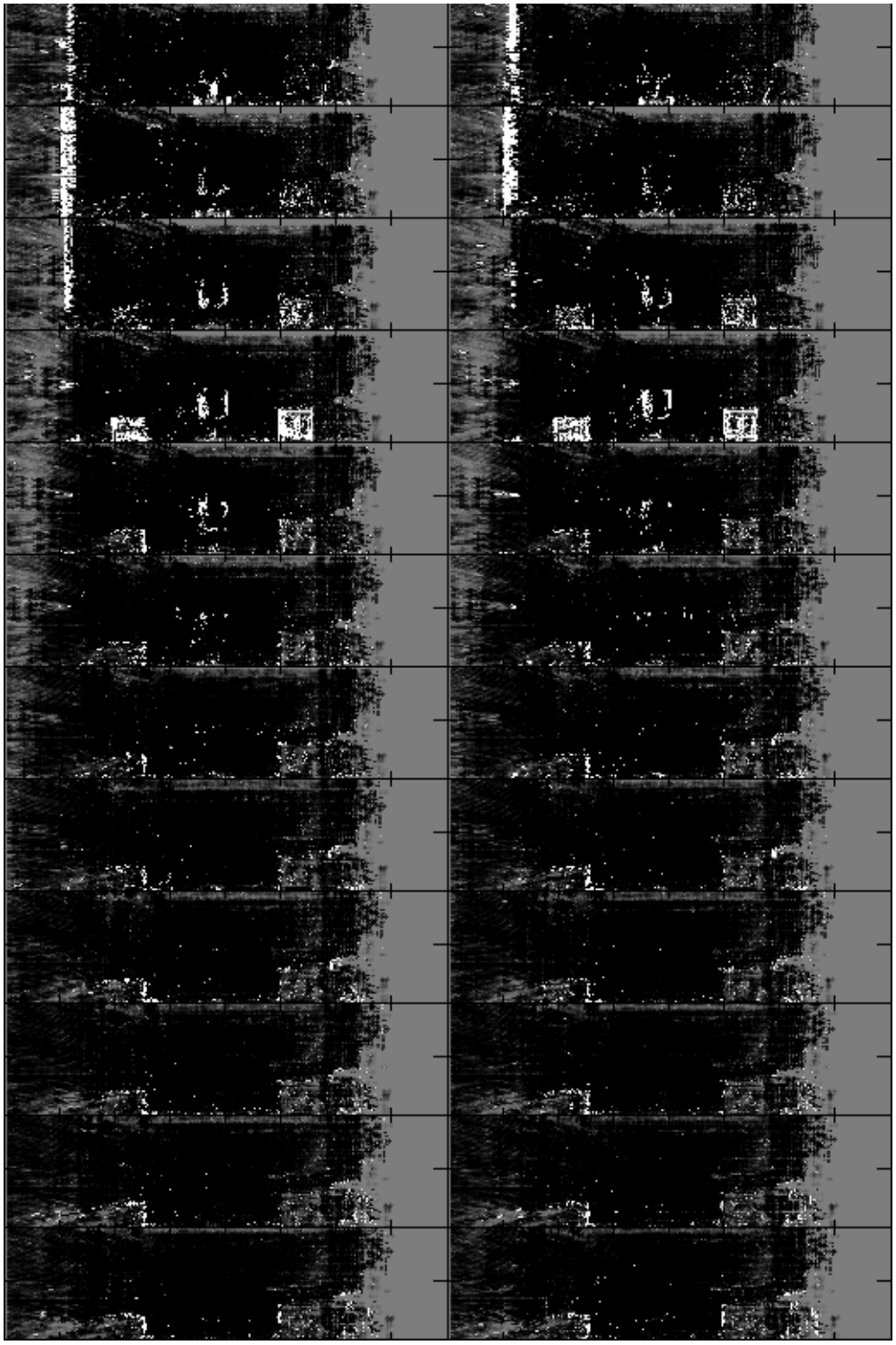


FIGURE 21 *Vertical slices marching towards the front of the room, oriented as if viewed from the front looking backward. Slices of the chair and small box seen in figure 18 top are visible, as is a larger box, out of view in figure 18. The prominent vertical features running top to bottom in the first frames are cabinets that can just be made out on the left of figure 18 bottom.*

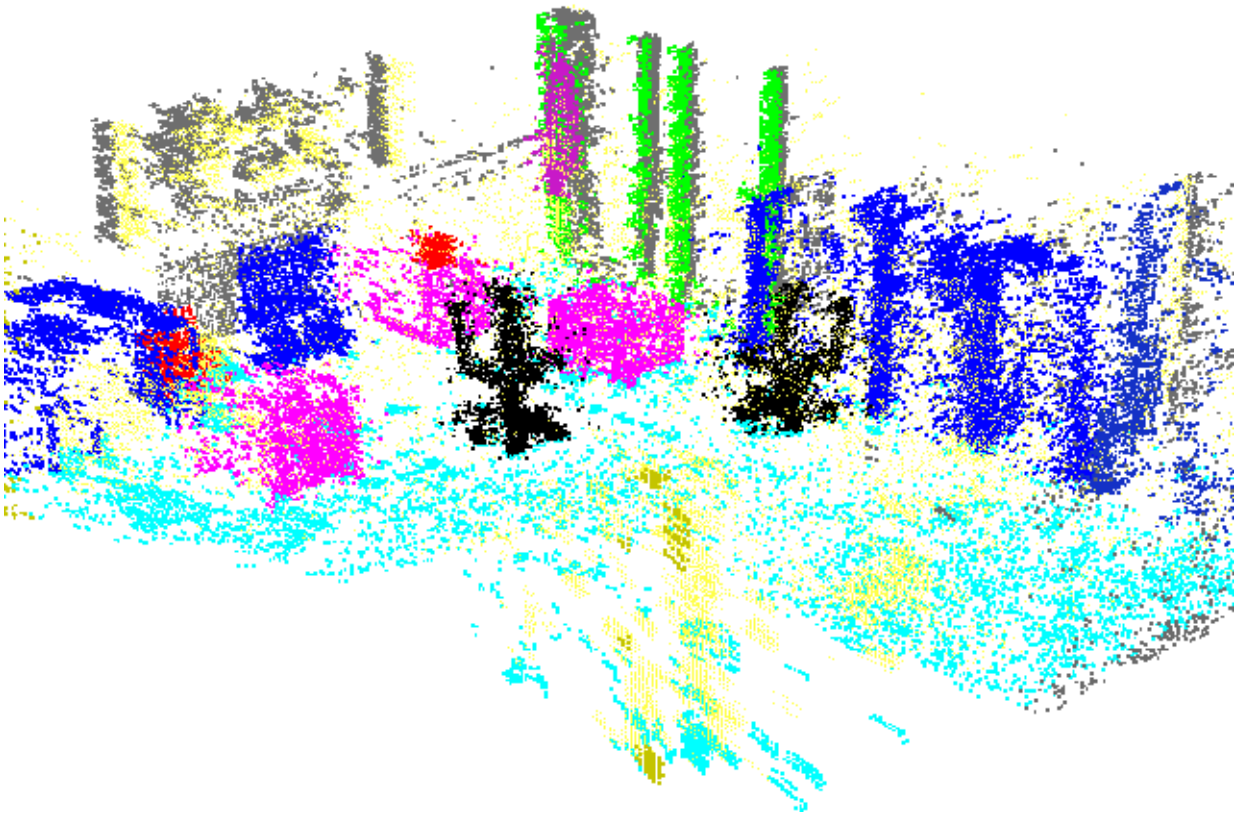


FIGURE 22 *A 2D projection of the 90,000 occupied cells in the 4 million cell 3D evidence grid of the laboratory. About a dozen box-shaped regions have been spotlighted in color for clarity. (Apologies to readers with monochrome versions.)*

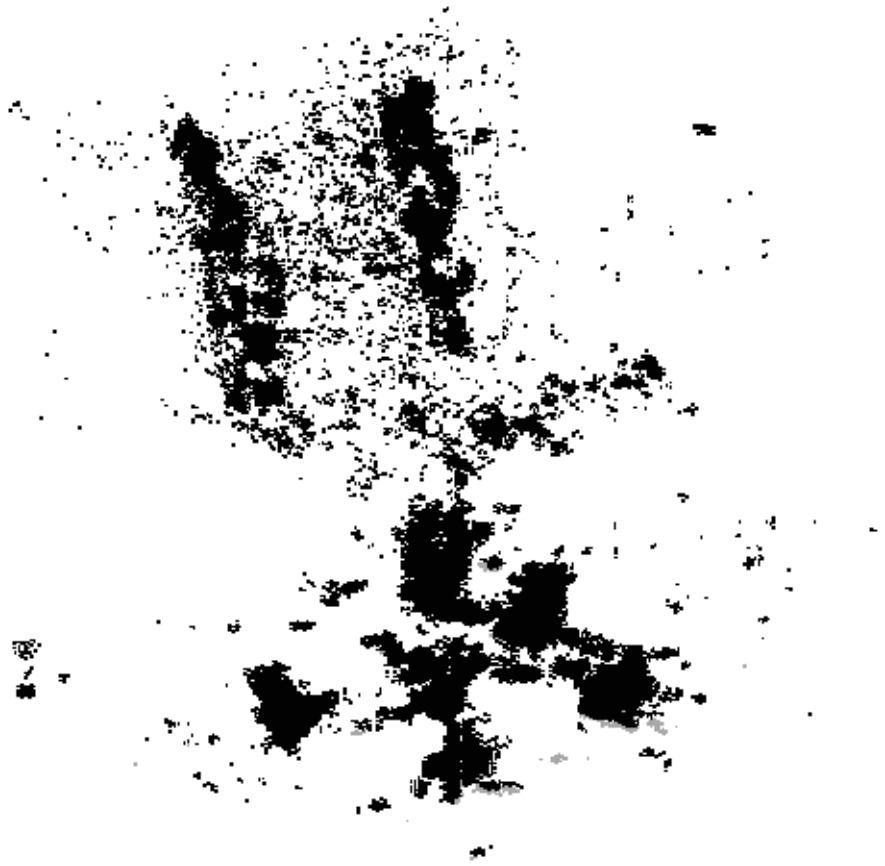


FIGURE 23 *A 2D projection of the 10,000 occupied cells in the high resolution grid of the chair.*

17 References

[Moravec81]

Author Moravec, Hans P. Robotics Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA
Title Robot rover visual navigation, Computer science. Artificial intelligence; no. 3
Source Ann Arbor, Mich. : UMI Research Press, c1981.

Abstract The Stanford AI Lab cart is a remotely controlled TV equipped mobile robot. A computer program has driven the cart through cluttered spaces, gaining its knowledge of the world entirely from images broadcast by the onboard TV system. The cart uses several kinds of stereo to locate objects around it in 3D and to deduce its own motion. It plans an obstacle avoiding path to a desired destination on the basis of a model built with this information. The plan changes as the cart perceives new obstacles on its journey. The system is reliable for short runs, but slow. The cart moves one meter every ten to fifteen minutes, in lurches. After rolling a meter it stops, takes some pictures and thinks about them for a long time. Then it plans a new path, executes a little of it, and pauses again. It has successfully driven the cart through several 20 meter courses (each taking about five hours) complex enough to necessitate three or four avoiding swerves. Some weaknesses and possible improvements were suggested by these and other, less successful, runs. Revision of the author's thesis (Ph.D.)--Stanford, 1980. Includes index. Bibliography: p. 149-150.

[Moravec83]

Author Moravec, Hans P.; Robotics Institute, Carnegie Mellon University, PA, USA
Title The Stanford Cart and the CMU Rover
Source Source:Proceedings of the IEEE; vol.71, no. 7; Jul. 1983 pp.; pp. 872-13pp.

Abstract The Stanford Cart was a remotely controlled TV equipped mobile robot. A computer program was written which drove the Cart through cluttered spaces, gaining its knowledge of the world entirely from images broadcast by an onboard TV system. The CMU Rover is a more capable, and nearly operational, robot being built to develop and extend the Stanford work and to explore new directions. The Cart used several kinds of stereoopsis to locate objects around it in 3D and to deduce its own motion. It planned an obstacle avoiding path to a desired destination on the basis of a model built with this information. The plan changed as the Cart perceived new obstacles on its journey. The system was reliable for short runs, but slow. The Cart moved one meter every ten to fifteen minutes, in lurches. After rolling a meter it stopped, took some pictures and thought about them for a long time. Then it planned a new path, executed a little of it, and paused again. It successfully drove the Cart through several 20 meter courses (each taking about five hours) complex enough to necessitate three or four avoiding swerves; it failed in other trials in revealing ways. The Rover system has been designed with maximum mechanical and control system flexibility to support a wide range of research in perception and control. It features an omnidirectional steering system, a dozen onboard processors for essential real time tasks, and a large remote computer to be helped by a high speed digitizing / data playback unit and a high performance array processor. Distributed high level control software similar in organization to the Hearsay II speech understanding system and the beginnings of a vision library are being readied. By analogy with the evolution of natural intelligence, we believe that incrementally solving the control and perception problems of an autonomous mobile mechanism is one of the best ways of arriving at general artificial intelligence.

[Moravec-Elfes85]

Author Moravec, H.P.; Elfes, A.E.; Robotics Institute, Carnegie Mellon University, PA, USA
Title High Resolution Maps from Wide Angle Sonar
Source Proceedings of the 1985 IEEE International Conference on Robotics and Automation, St. Louis, March, 1985, pp 116-121;

Abstract We describe the use of multiple wide-angle sonar range measurements to map the surroundings of an autonomous mobile robot. A sonar range reading provides information concerning empty and occupied volumes in a cone (subtending 30 deg in our case) in front of the sensor. The reading is modelled as probability profiles projected onto a rasterized map, where somewhere occupied and everywhere empty areas are represented. Range measurements from multiple points of view (taken from multiple sensors on the robot, and from the same sensors after robot moves) are systematically integrated in the map. Overlapping empty volumes re-inforce each other, and serve to condense the range of occupied volumes. The map definition improves as more readings are added. The final map shows regions probably occupied, probably unoccupied, and unknown areas. The method deals effectively with clutter, and can be used for motion planning and for extended landmark recognition. This system has been tested on the Neptune mobile robot at CMU.

[Elfes87]

Author Elfes, A.; Robotics Inst., Carnegie-Mellon Univ., Pittsburgh, PA, USA
Title Sonar-based real-world mapping and navigation
Source IEEE Journal of Robotics and Automation; IEEE J. Robot. Autom. (USA); vol.RA-3, no.3; A07; June 1987 ; pp. 249-65 pp.

Abstract A sonar-based mapping and navigation system developed for an autonomous mobile robot operating in unknown and unstructured environments is described. The system uses sonar range data to build a multileveled description of the robot's surroundings. Sonar readings are interpreted using probability profiles to determine empty and occupied areas. Range measurements from multiple points of view are integrated into a sensor-level sonar map, using a robust method that combines the sensor information in such a way as to cope with uncertainties and errors in the data. The sonar mapping procedures have been implemented as part of an autonomous mobile robot navigation system called Dolphin. The major modules of this system are described and related to the various mapping representations used. Results from actual runs are presented, and further research is mentioned. The system is also situated within the wider context of developing an advanced software architecture for autonomous mobile robots.

[Moravec-Blackwell93]

Author Moravec, H.P.; Blackwell, M.; Robotics Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA
Title Learning Sensor Models for Evidence Grids
Source CMU Robotics Institute 1991 Annual Research Review, 1993, pp.8-15.

Abstract Evidence grids (aka. occupancy, probability and certainty grids) are a probabilistic, finite-element representation of robot spatial knowledge. The grids allow the efficient accumulation of small amounts of information from individual sensor readings into increasingly accurate and confident maps. Each sensor measurement is translated, via a sensor model, into a spatial evidence distribution that is added to a grid representing the robot's surroundings. In our first applications of the method, on a mobile robot with a ring of 24 Polaroid sonar transducers autonomously navigating a cluttered room, we constructed the sensor model from a cursory examination of the Polaroid literature. Despite the ad-hoc

model, the grid approach worked far better than an older program, a decade in development, that used a geometric model. It successfully navigated cluttered rooms, most hallways, a coal mine and outdoors. The original program failed in a smooth-walled narrow corridor, where most sonar pulses, deflected by mirrorlike walls, indicated overlong ranges. The evidence grid method might be able to slowly accumulate evidence from such data, if only the sensor model accurately represented the modest information contained in a reading, as our ad-hoc model did not. This paper reports on a learning program that finds good models automatically. The sensor model is formulated as a closed form expression shaped by several parameters. The parameters are adjusted, in a hill-climbing process, that maximizes the match between a hand-constructed ideal map and a map built by the model with data from a robot test run in the mapped area. Using this approach with a 9- parameter function a program using several weeks of Sparc1+ workstation search time was able to produce a crisp, correct map of the difficult smooth hallway, from data that produces an unrecognizable splatter when interpreted by our original ad-hoc sensor model.

[Hellwich&al94]

Author Hellwich, O.; Heipke, C.; Liang Tang; Ebner, H.; Mayr, W.; Tech. U. Munchen, Germany
Title Experiences with automatic relative orientation
Source ISPRS Commission III Symposium Spatial Information from Digital Photogrammetry and Computer Vision; Munich, Germany; 5-9 Sept. 199 Sponsored by: SPIE; Proceedings of the SPIE - The International Society for Optical Engineering; Proc. SPIE - Int. Soc. Opt. Eng. (USA); vol.2357, pt.1; 1994; pp. 370-8

Abstract A report on recent experiences with a new procedure for automatic relative orientation is given. A hierarchical approach provides conjugate points using image pyramids. The implemented method is based on feature extraction by an interest operator, feature matching between the two images, area based image correlation, a robust least squares bundle adjustment and feature tracking through several levels of the image pyramid. The automatic procedure proved to be successful for various combinations of terrain relief surface cover and image scale. The paper presents the results of intensive tests. It was found that only a limited number of control parameters of the algorithm has to be individually adjusted to the terrain specifics. Further investigations need to be conducted to find out whether the project dependent parameters can be reliably predicted for certain classes of images.

[Stunz-Knopfle-Roth94]

Author Strunz, G.; Knopfle, W.; Roth, A.; Remote Sensing Data Centre, German Aerosp. Res. Establ., Oberpfaffenhofen, Germany
Title Automation of tie pointing procedure for the geocoding of satellite images
Source ISPRS Commission III Symposium Spatial Information from Digital Photogrammetry and Computer Vision; Munich, Germany; 5-9 Sept. 1994, SPIE; Proceedings of the SPIE - The International Society for Optical Engineering; Proc. SPIE - Int. Soc. Opt. Eng. (USA); vol.2357, pt.2; 1994; pp. 793-800

Abstract The comprehensive utilization of the information content provided by remote sensing satellites requires the multitemporal and multisensoral analysis of the image data. Prior to this analysis, processing is necessary to correct for geometric distortions. To handle the large amount of data and the high data rates, the traditional approach of visual identification of control or tie points is not an acceptable solution. The paper describes two developments at the German Remote Sensing Data Centre, which aim at the automatic generation of tie points for optical sensors and radar images, and the operational utilization of automatic tie pointing for ERS-1 SAR data. A concept for the multitemporal registration of sat-

ellite images from optical sensors is described. This approach comprises the detection of distinct points by a so-called interest operator in one image. By cross correlation, the corresponding points in the other images are traced. Subpixel accuracy is achieved by a least squares technique for digital image matching. Practical results are shown, where multitemporal image data from the Landsat TM sensor are used. The paper also describes the operational utilization of an automatic tie pointing which has been realized for the geocoding of the ERS-1 SAR data at the German Processing and Archiving Facility. Significant features are extracted from reference data sets. The procedure is described as well as the accuracy achieved. An outlook is given concerning several automatic approaches being investigated.

[Koenig-Simmons96]

Author Koenig, S.; Simmons, R.G.; Sch. of Comp. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA

Title Unsupervised learning of probabilistic models for robot navigation

Source Proceedings of IEEE International Conference on Robotics and Automation; Part: vol.3; Minneapolis, MN, USA; 22-28 April 1996; sponsored by: IEEE Robotics & Autom. Soc; Proceedings. 1996 IEEE International Conference on Robotics and Automation (Cat. No. 96CH35857); New York, NY, USA; IEEE; 4 vol. lxiv+3749; 1996; pp. 2301-8 vol.3

Abstract Navigation methods for office delivery robots need to take various sources of uncertainty into account in order to get robust performance. In previous work, we developed a reliable navigation technique that uses partially observable Markov models to represent metric, actuator and sensor uncertainties. This paper describes an algorithm that adjusts the probabilities of the initial Markov model by passively observing the robot's interactions with its environment. The learned probabilities more accurately reflect the actual uncertainties in the environment, which ultimately leads to improved navigation performance. The algorithm, an extension of the Baum-Welch algorithm, learns without a teacher and addresses the issues of limited memory and the cost of collecting training data. Empirical results show that the algorithm learns good Markov models with a small amount of training data.

[Yamauchi96]

Author Yamauchi, B.; Inst. for the Study of Learning & Experience, Palo Alto, CA, USA

Title Mobile robot localization in dynamic environments using dead reckoning and evidence grids

Source Proceedings of IEEE International Conference on Robotics and Automation; Part: vol.2; Minneapolis, MN, USA; 22-28 April 1996; Sponsored by: IEEE Robotics & Autom. Soc; Proceedings. 1996 IEEE International Conference on Robotics and Automation (Cat. No. 96CH35857); New York, NY, USA; IEEE; 4 vol. lxiv+3749; 1996; pp. 1401-6 vol.2

Abstract Dead reckoning provides a simple way to keep track of a mobile robot's location. However, due to slippage between the robot's wheels and the underlying surface, this position estimate accumulates errors over time. In this paper, we introduce a method for correcting dead reckoning errors by matching evidence grids constructed at different times. A hill-climbing algorithm is used to search the space of possible translations and rotations used to transform one grid into the other. The transformation resulting in the best match is used to correct the robot's position estimate. This technique has been tested on a real mobile robot and has demonstrated robustness to transient changes (moving people) and lasting changes (rearranged obstacles) in dynamic environments.

[Schultz-Adams-Grefenstette96]

Author Alan C. Schultz, William Adams, and John J. Grefenstette (1996). NCARAI,Naval Research Laboratory, Washington, DC 20375-5337

Title Continuous localization using evidence grids

Source NCARAI Report AIC-96-007, Naval Research Laboratory, Washington, DC. schultz@aic.nrl.navy.mil

Abstract The evidence grid representation provides a uniform representation for fusing temporally and spatially distinct sensor readings. However, the use of evidence grids requires that the robot be localized within its environment. Odometry errors typically accumulate over time, making localization estimates degrade, and introducing significant errors into evidence grids as they are built. We have addressed this problem by developing a new method for ``continuous localization'', in which the robot corrects its localization estimates incrementally and on the fly. Assuming the mobile robot has a map of its environment represented as an evidence grid, localization is achieved by building a series of ``local perception grids,'' also represented as evidence grids, based on localized sensor readings and the current odometry, and then registering the local and global grids. The registration produces an offset which is used to correct the odometry. Results are given on the effectiveness of this method, and quantify the improvement of continuous localization over dead reckoning. Further results show that maps built of the room using evidence grids while the robot is performing continuous localization show no appreciable systematic errors due to odometric error; it appears that maps of the room can simultaneously be learned and used for continous localization.

[Martin-Moravec96]

Author Author:Martin, M.C.; Moravec, H.P.; Robotics Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA

Title Robot Evidence Grids

Source Carnegie Mellon University, Robotics Institute Technical Report CMU-RI-TR-96-06

Abstract The evidence grid representation was formulated at the CMU Mobile Robot Laboratory in 1983 to turn wide angle range measurements from cheap mobile robot-mounted sonar sensors into detailed spatial maps. Whereas most older approaches to mapping considered the location of objects, the grid approach weighs the “objectness” of locations. It accumulates diffuse evidence about the occupancy of a grid of small volumes of nearby space from individual sensor readings into increasingly confident and detailed maps of a robot’s surroundings. It worked surprisingly well in first implementation for sonar navigation in cluttered rooms. In the past decade its use has been extended to range measurements from stereoscopic vision and other sensors, sonar in very difficult specular environments, and other contexts. The most dramatic extension yet, from 2D grid maps with thousands of cells to 3D grids with millions, is underway. This paper presents the mathematical and probabilistic framework we now use for evidence grids. It gives the history of the grid representation, and its relation to other spatial modeling approaches. It discusses earlier formulations and their limitations, and documents several extensions. A list of open issues and research topics is then presented, followed by a literature survey.