

Visual Mapping by a Robot Rover

Hans P. Moravec
AI Lab, Computer Science Dept.
Stanford University, Stanford, Ca, 94305

Introduction

The real world provides potentially huge quantities of noisy data. The lower levels of the perception process must extract reliable and meaningful measurements from this sullied wealth. Much existing work in vision evades the bulk of the noise problem, either by physically optimizing the nature, lighting and coloration of the viewed scene, or by starting with very high quality digitized photography.

This paper is about a situation in which the noise could not be ignored. The routines described extract very reliable three dimensional information from unconstrained scenes perceived by an imprecise vehicle transmitting pictures whose quality has caused some vision researchers to hold their (metaphorical) noses.

The cart's noise problems may be unusually severe, but are present to some degree in all real world sensors and effectors. For this reason our solutions may be generally useful as part of the solid foundations of ambitious systems well beyond robot explorers, as computer vision moves from optimized workspaces into the inherently dirty real world.

The underlying theme of the work is the use of redundancy (which the real world supplies in profusion) to counter noise. In many cases I was forced into this by performance failure of other approaches.

Setting

The Cart [1] is a rickety camera equipped rover, computer controlled over a radio link. This report describes a working program that has successfully used pictures obtained from the cart as it was driven, unguided, through cluttered real world environments to obtain navigational information and to build correct three dimensional maps of its surroundings and to plan obstacle avoiding paths to desired locations. The program builds a sparse 3D map of its surroundings, but makes no attempt to classify the objects it sees, except as obstacles or non-obstacles. Work underway will enable the program to actively keep the cart on the planned path.

In 1977 I abandoned as unworkable an approach [2] that tried to obtain depth information from a scene exclusively through motion stereo pairs. The matching errors in our very noisy pictures, combined with an uncertain baseline set by vehicle motion, often caused incorrect measurements from the routine which deduced vehicle motion from the perceived shift of features in the scene. The navigational model quickly became hopelessly inaccurate. Several attempts at software fixes proved fruitless.

New hardware, to extract more noisy data from the world, has overcome the problem. The cart is now equipped with a mechanism that uses a stepping motor to slide the TV camera horizontally on a 52 cm track in precise 0.25 mm steps. The cart system uses this arrangement to obtain nine views of the scene, at 6.4 cm intervals. These provide a highly redundant nine-eyed stereo baseline for reliable distance measurements even in the presence of much noise.

Overview

Mapping runs occasionally begin with a camera calibration step (at other times old calibrations are used). The cart is placed 3 meters in front of and pointed towards a 3 meter square pattern of dots on a wall. A program digitizes the pattern, and automatically locates the spots and determines the camera's focal length as well as distortions.

The cart is then placed on a clear path in a cluttered space (environments have included a large room cluttered with boxes and old machinery, and an outdoor roadway with parked cars, trees and signposts). A navigating program is begun by reading in a calibration file. It then slides the camera to the left limit of its track. A TV image is digitized, the camera then slides 6.4 cm to the right, another picture is taken, and so on for a total of nine images. The cart sits immobile for about ten real time minutes while our timeshared KL-10 works on the pictures for 180 compute seconds. Then it lurches about a meter forward, stops, and repeats the camera motion.

As the cart moves slowly forward, meter by meter, the program generates and updates a three dimensional world model of features it has seen, and deduces the vehicle's motion through them. This model is displayed in map form. Initially the map is blank. As time (and the cart) proceeds more and more points are added. The map includes the cart's past trajectory, and a path to a desired destination that avoids all known obstacles.

The program is being extended to issue steering commands, intended to keep the cart on its planned obstacle avoiding path, before each lurch.

Digitizing (Temporal Redundancy)

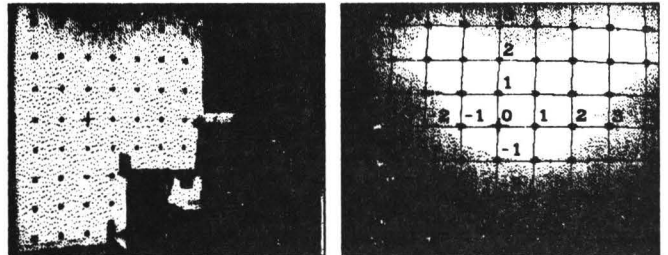
Digitizing images over a sometimes noisy radio link has its problems, but gaining adequate images is a crucial part of the cart system.

The picture is transmitted at 524 MHz, a wavelength of 57 cm. Reflections from nearby objects sometimes cancel the signal to a receiver. The programs are able to get their picture through any one of 4 TV receivers. They measure scanline to scanline noise on each in turn, and then pick the one with lowest noise. In future I may simply switch antennas to a single receiver.

Our decade old digitizer extracts a picture 280 samples wide by 240 high by 4 bits/sample from broadcast standard video. Four bits is only sixteen gray levels, and sometimes the noise level is high. Summing a number of frames can increase the intensity resolution and average out noise, but introduces new problems. The sync detector on the digitizer sometimes fails, especially when the signal is noisy, and the picture jitters or "rolls". Adding improperly registered pictures results in blurring. The problem has been solved by a program that digitizes a large number of frames (about 30), and by an efficient sampling of the pixels intercompares them to find the subset of pictures that are most nearly alike. These are then combined. Since most images have not suffered sync loss, the result is almost always a tolerable six bit picture.

Picture Reduction (Area Redundancy)

The cart vision programs make extensive use of reduced versions of input images. Digitized pictures are compressed repeatedly by linear factors of two, four pixels being added to become one, until the whole image has shrunk to a single pixel. The original image and all of its reductions are available to all subsequent steps.



The cart before its calibration array, and the digitized spot pattern, overlaid by an ideal grid distorted by the calibration polynomial.

Calibration (Redundancy in Regularity)

The cart camera's image is distorted by the details of its lens, and of its vidicon's scan. Its precision has been enhanced by an automatic focal length and distortion determining program.

The cart is parked a precise distance in front a square pattern of spots around a central cross. The program digitizes an image of the array, locates the spots and the cross, and constructs a 2D polynomial that relates the position of the spots in the image to their position in an ideal unity focal length camera, and another polynomial that converts points from the ideal camera to points in the image. These polynomials are used to correct the positions of perceived objects in later scenes.

The algorithm begins by determining its approximate spacing and orientation of the spot array. It trims the picture to make it square, reduces it by averaging to 64 by 64, calculates the Fourier transform of the reduced image and takes its power spectrum, arriving at a 2D transform symmetric about the origin, and having strong peaks at frequencies corresponding to the horizontal, vertical and half diagonal spacing, with weaker peaks at the harmonics. It multiplies each point $[i, j]$ in this transform by point $[-j, i]$ and points $[j-i, j+i]$ and $[i+j, j-i]$, effectively folding the primary peaks onto one another. The strongest peak in the 90 degree wedge around the y axis gives the spacing and orientation information needed by the next part of the process.

The directional variance interest operator described later is applied to roughly locate a spot near the center of the image. A special operator examines a window surrounding this position, generates a histogram of intensity values within the window and selects a threshold for separating the black spot from the

white background. It calculates the centroid and verifies the spot's shape by measuring the first and second moment. The region of found spots is grown outward from this seed, using this operator guided by the Fourier determined orientation/spacing parameters.

The Fourier numbers are also used to construct a template for the expected appearance of the cross in the middle of the array. Each of the found spots is matched to this cross template, and the closest match in the central portion of the picture is declared to be the origin.

Two fourth degree least squares polynomials in two variables relating the actual positions of the spots to the ideal positions are generated and written into a file.

The program tolerates a wide range of spot parameters (about 3 to 12 spots across), arbitrary image rotation, and is very robust. It has worked without error on over 50 images. The test pattern for the cart is a 3 meter square painted on a wall, with 5 cm spots at 30 cm intervals. The program has also been used successfully with a small array (22 x 28 cm) to calibrate cameras other than the cart's.

Choosing Features (Redundancy in Boundaries)

The cart vision code deals with very simple primitive entities, localized regions called features. A feature is conceptually a point in the three dimensional world, but it is found by examining localities larger than points in pictures. A feature is good if it can be located unambiguously in different views of a scene. A uniformly colored region or a simple edge does not make for good features because its parts are indistinguishable. Regions, such as corners, with high contrast in orthogonal directions are best.

New features in images are picked by a subroutine called the *Interest operator*. It tries to select a relatively uniform scattering, to maximize the probability that a few features will be picked on every visible object, and tries to choose areas that can be easily found in other images. Both goals are achieved by returning regions that are local maxima of a directional variance measure.

Directional variance is measured over small square windows. Sums of squares of differences of pixels adjacent in each of four directions (horizontal, vertical and two diagonals) over each window are calculated, and the window's interest measure is the minimum of these four sums.

Features are chosen where the interest measure has local maxima. The feature is conceptually the point at the center of the window with this locally maximal value.

The effects of noise are alleviated and the processing time is shortened by applying the operator to a reduced image. In the current program original images are 240 lines high by 256 pixels wide. The interest operator is applied to the 120 by 128 version, on windows 3 pixels square.

Once a feature is chosen, its appearance is recorded as series of excerpts from the reduced image sequence. A 6x6 window is excised around the feature's location from each of the variously reduced pictures. Only a tiny fraction of the area of the original (unreduced) image is extracted. Four times as much of the x2 reduced image is stored, sixteen times as much of the x4 reduction, and so on until at some level we have the whole image. The final result is a series of 6 by 6 pictures, beginning with a very blurry rendition of the whole picture, gradually zooming in linear expansions of two to a sharp closeup of the feature.

Finding Features Again (Redundancy in Continuity)

Deducing the 3D location of features from their projections in 2D images requires that we know their position in two or more such images.

The correlator is a subroutine that takes a feature description (such as the interest operator produces), and finds the best match in a different image. Its search area can be an entire new picture, or a rectangular sub-window.

The search uses a coarse to fine strategy that begins in reduced versions of the pictures. Typically the first step takes place at the x16 (linear) reduction level. The 6 by 6 window at that level in the feature description, which covers about one seventh of the total area of the original picture, is convolved with the search area in the correspondingly reduced version of the second picture. The 6 by 6 description patch is moved pixel by pixel over the approximately 15 by 16 destination picture, and a correlation coefficient is calculated for each trial position.

The position with the best match is recorded. The 6x6 area it occupies in the second picture is mapped to the x8 reduction level, where the corresponding region is 12 pixels by 12. The 6 by 6 window in the x8 reduced level of the feature description is then convolved with this 12 by 12 area, and the position of best match is recorded and used as a search area for the x4 level.

The process continues, matching smaller and smaller, but more and more detailed windows until a 6 by 6 area is selected in the unreduced picture.

The work at each level is about the same, finding a 6 by 6 window in a 12 by 12 search area. It involves 49 summations of 36 quantities. In our example there were 5 such levels.

Slider Stereo (Spatial Redundancy)

At each pause on its computer controlled itinerary the cart slides its camera from left to right on the 52 cm track, taking 9 pictures at precise 6.5 cm intervals.

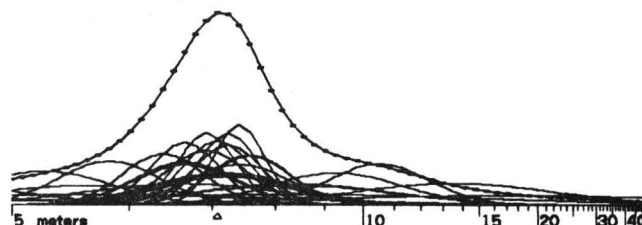
Points are chosen in the fifth (middle) of these 9 images, either by the correlator, to match features from previous positions, or by the interest operator.

The camera slides parallel to the horizontal axis of the (distortion corrected) camera co-ordinate system, so the parallax induced apparent displacement of features from frame to frame in the 9 pictures is purely in the X direction.

The correlator looks for the selected features from the central image in each of the eight other pictures. The search is restricted to a narrow horizontal band. This has little effect on the computation time, but it reduces the probability of incorrect matches.

In the case of correct matches, the distance to the feature is inversely proportional to its displacement from one image to another. The uncertainty in such a measurement is the difference in distance a shift one pixel in the image would make. The uncertainty varies inversely with the physical separation of the camera positions where the pictures were taken (the stereo baseline).

After the correlation step the program knows a feature's position in nine images. It considers each of the 36 possible image pairings as a stereo baseline, and records the estimated distance to the feature in a histogram of inverse distances. Each measurement adds a little normal curve to the histogram, with mean at the estimated distance, and standard deviation inversely proportional to the baseline, modelling the distance uncertainty. The area under the each curve is made proportional to the product of the correlation coefficients of the matches in the two images (in central image this coefficient is taken as unity), reflecting the confidence that the correlations were correct. The area is also scaled by the normalized dot products of X axis and the shift of the features in each of the two baseline images from the central image. That is, a distance measurement is penalized if there is significant motion of the feature in the Y direction.



Results of a nine-eyed stereo ranging on a single feature. The small curves are from individual image pairs, the beaded curve is the final histogram. The feature is 7.1 meters distant.

The distance to the feature is indicated by the largest peak in the resulting histogram, if this peak is above a certain threshold. If below, the feature is forgotten about.

This adding of normal curves and locating the peak in the sum is an efficient way of finding the strongest cluster of mutually agreeing measurements. The distance measurements from different pairings of incorrect matches are usually inconsistent. When the normal curves from 36 pairs are added up, the correct matches agree with each other, and build up a large peak in the histogram, while incorrect matches spread themselves thinly. Two or three correct correlations out of the eight will usually build a peak sufficient to offset a larger number of errors.

In this way eight applications of a mildly reliable operator interact to make a very reliable distance measurement.

Motion Stereo (Redundancy of Rigid Motions)

The cart navigates exclusively by vision. It deduces its own motion from the apparent 3D shift of the features around it.

After having determined the 3D location of objects at one position, the computer drives the cart about a meter forward.

At the new position it slides the camera and takes nine pictures. The correlator is applied in an attempt to find all the features successfully located at the previous position. Feature descriptions extracted from the central image at the last position are searched for in the central image at the new stopping place.

Slider stereo then determines the distance of the features so found from the cart's new position. The program now knows the 3D position of the features relative to its camera at the old and the new locations. It can deduce its own movement by finding the 3D co-ordinate transform that relates the two.

There can be mis-matches in the correlations between the central images at two positions. Before the cart motion is deduced, the feature positions are checked for consistency. Although it doesn't yet have the co-ordinate transform between the old and new camera systems, the program knows the distance between pairs of positions should be the same in both. It makes a matrix in which element $[i,j]$ is the absolute value of the difference in distances between points i and j in the first and second co-ordinate systems divided by the expected error (based on the one pixel uncertainty of the ranging).

Each row of this matrix is summed, giving an indication of how much each point disagrees with the other points. The idea is that while points in error

disagree with virtually all points, correct positions agree with all the other correct ones, and disagree only with the bad ones.

The worst point is deleted, and its effect is removed from the remaining points in the row sums. This pruning is repeated until the worst error is within the error expected from the ranging uncertainty.

After the pruning, the program has a number of points, typically 10 to 20, whose position error is small and known. These are recorded in a world model.

From the 3D camera co-ordinates of these sanitized points at both positions, the program finds a three dimensional rotation and translation that, if applied to the co-ordinates of the features at the first position, minimizes the sum of the squares of the distances between the transformed first co-ordinates and the raw co-ordinates of the corresponding points at the second position, each term divided by the square of the *a priori* expected error.

The error expression is expanded. It becomes a function of the rotation and translation, with parameters that are the weighted averages of the x, y and z co-ordinates of the features at the two positions, and averages of their various cross-products. These averages need to be determined only once, at the beginning of the transform finding process.

To minimize the error expression, its partial derivative with respect to each variable is set to zero. It is easy to simultaneously solve the three linear equations thus resulting for the vector offset, getting the optimal offset values for a general rotation. This gives symbolic expressions (linear combinations of the rotation matrix coefficients) for each of the three vector components. Substituting these values into the error expression makes it a function of the rotation alone. This reduced error expression is used in all the subsequent steps.

Minimizing the error expression under rotation is surprisingly difficult, mainly because of the non-linear constraints in the 3D rotation matrix.

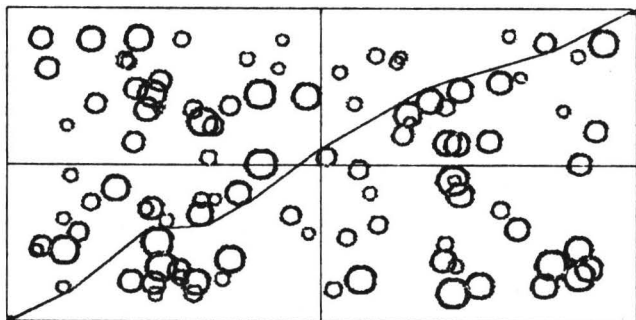
The program begins by ignoring the non-linearities. It solves for the general 3D linear transformation, nine elements of a matrix, that minimizes the least square error. The derivatives of the error expression with respect to each of the matrix coefficients are equated to zero, and the nine resulting simultaneous linear equations are solved for the nine coefficients. If the points had undergone an error-free rigid rotation and translation between the two positions, the result would be the desired rotation matrix, and the problem would be solved.

Because there are errors in the determined position of the features, the resulting matrix is usually not simply a rotation, but involves stretching and skewing. The program ortho-normalizes the matrix, and converts its 9 independent linear elements to unambiguously non-linear expressions in three parameters that uniquely characterize a rotation.

This new error expression is differentiated with respect to each of the three rotation parameters, and the resulting expressions are equated to zero, giving us three non-linear equations in three unknowns. These are solved by Newton's method.

Four or five iterations usually brings the parameters to within our floating point accuracy of the correct values. In the occasional case of non-convergence, the program picks a new starting point, and tries again, up to 100 times. The rotation with the smallest least square error ever encountered during such a search is returned as the answer.

Since the summations over the co-ordinate cross-products are done once and for all at the beginning of the transformation determination, each iteration, involving evaluation of about a dozen moderately large expressions and a 3 by 3 matrix inversion, is relatively fast. The whole solving process, even in cases of pathological non-convergence, takes one or two seconds of computer time.



The path planner at work. Point of origin is lower left, destination is upper right. Obstacles are indicated by circles. This field was randomly generated. The apparent radius of the obstacles in a real case would be augmented by the cart's radius.

Path Planning

The features in the cart's 3D world model can be thought of as fuzzy ellipsoids, whose dimensions reflect the program's uncertainty of their position. Repeated applications of the interest operator as the cart moves cause virtually

all visible objects to be become modelled as clusters of overlapping ellipsoids.

To simplify the problem, the ellipsoids are approximated by spheres. Those spheres sufficiently above the floor and below the cart's maximum height are projected on the floor as circles. The cart itself is modelled as a 3 meter sphere. The path finding problem then becomes one of maneuvering the cart's 3 meter projected circle between the (usually smaller) circles of the potential obstacles to a desired location.

It is convenient (and equivalent) to conceptually shrink the cart to a point, and add its radius to each and every obstacle. An optimum path in this environment will consist of either a straight run between start and finish, or a series of tangential segments between the cylinders and contacting arcs (imagine loosely laying a string from start to finish between the cylinders, then pulling it tight).

There are four possible paths between each pair of obstacles, because each tangent can approach clockwise or counterclockwise. The program finds a near optimal obstacle avoiding route to the desired goal in $O(x^3)$ time, using an extension of a shortest path in a graph algorithm first given by Dijkstra [3]. Because our tangent space cannot be modelled exactly as a simple graph, because of the route dependent arc lengths on the circular part of the paths, the program finds good paths, but not always the best.

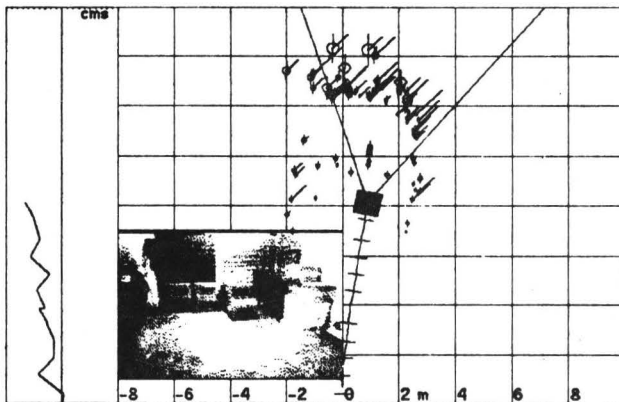
Results

Because of a hardware problem with the steering electronics, the program has thus far driven the cart without steering control, in nearly straight lines only. During these runs it has built correct world maps, and planned (but not executed) good obstacle avoiding paths.

The longest run thus far has been for 14 nine-picture sets, a total distance of about 10 meters. During this run, made in a large cluttered room, the program inserted 120 features into its world model, of which about half were potential obstacles, most of the other half being marks on the floor. I was able to find no errors beyond the known position uncertainties in the resulting map. The parts of the room that were in the cart's field of view for the bulk of the run are densely represented, with no possible paths falsely seeming to be obstacle-free. Several shorter runs gave similarly robust results.

The steering problem is being repaired, and I expect a long outdoor run, in which the cart will maneuver around several obstacles, in the near future.

A longer description of this work is available from the author.



Cart's world model at the end of a 14 lurch run. Photo is view from final position. Features are marked by circles, indicating position uncertainty, on the ground. 45 degree lines growing out of circles are height of features above ground. Cart's position is marked by black square. Forward pointing rays are its field of view. Tracks behind it are its itinerary, each stop is marked by a bar. Graph on left is its estimate of own height above ground. It shows the limiting accuracy (a few centimeters) of its self-position model.

References

1. R.A. Schmidt, "A Study of the Real-Time Control of a Computer Driven Vehicle", Stanford AI Memo 149, August 1971.
2. H.P. Moravec, "Towards Automatic Visual Obstacle Avoidance", *Proceedings of the 5th IJCAI*, MIT, Cambridge, Mass., 1977, p. 584.
3. E.W. Dijkstra, "A Note on Two Problems In Connection with Graphs," *Numerische Mathematik*, 1, 269-271 (1959).