Position Estimation for an Autonomous Mobile Robot in an Outdoor Environment

Raj Talluri and J. K. Aggarwal, *Fellow, IEEE*

Abstract— **This paper presents a solution to the position estimation problem of an autonomous land vehicle navigating in an unstructured mountainous terrain. A digital elevation map (DEM) of the area in which the robot is to navigate is assumed to be given. It is also assumed that the robot is equipped with a camera that can be panned and tilted, a compass, and an altimeter. No recognizable landmarks are assumed to be present in the environment in which the robot is to navigate, and the robot is not assumed to have an initial estimate of its position. The solution presented here makes use of the DEM information, and structures the problem as a constrained search paradigm by searching the DEM for the possible robot location. The shape and position of the horizon line in the image plane and the known camera geometry of the perspective projection are used as parameters to search the DEM. Geometric constraints are used to prune the search space significantly. The algorithm is made robust to errors in the imaging process by accounting for worst case errors. The approach is tested using real terrain data of areas in Colorado and Texas. The method is suitable for use in outdoor mobile robots and planetary rovers.**

I. Introduction

AUTONOMOUS mobile robots are one of the important areas of application of computer vision. The advantages of having a vehicle that can navigate without human intervention are many and varied, ranging from vehicles for use in hazardous industrial environments, battlefield surveillance vehicles, and planetary rovers.

The problems associated with navigating mobile robots in an indoor structured environment are reasonably well studied, and a number of different approaches have been suggested [1]–[6]. Outdoor navigation of a mobile robot in an unstructured environment is a more complex problem, and many issues remain open and unsolved. The DARPA ALV project [7], the CMU NAVLAB [8], and the University of Massachusetts mobile robot project [9] are among some of the significant research attempts in the area of outdoor navigation for autonomous vehicles.

Position estimation, or *spatial localization* as it is alternately called, is one of the primary requirements of any autonomous navigational task. The environment in which the robot is to navigate and the information available about the environment affect the problem significantly. The techniques for an indoor office type of environment are quite different from those called for in an outdoor scenario. Identifying landmarks, measuring range and/or attitude to these landmarks from the robot, and using this information to do positional estimation is one feasible way of spatial localization [10]–[13]. However, if the robot does not have the ability to detect any landmarks in the environment, these approaches do not work. The robot may be aided in its navigational task by a preloaded map of the environment. The problem to be solved in such a case is to establish a correspondence between the map and the images taken by the robot.

This paper considers the problem of an autonomous mobile robot navigating in an outdoor mountainous environment, equipped with a visual camera that can be panned and tilted. A digital elevation map (DEM) of the areas in which the robot is to navigate is provided to the robot. The robot is also assumed to be equipped with a compass and an altimeter to measure the altitude of the robot. In this paper, we present a method to estimate the position of the robot in such a scenario. Typical applications could be that of an autonomous land vehicle, such as a planetary rover. The approach presented formulates the position estimation problem as a constrained search problem. The horizon line contour (HLC) is extracted from the images and used to search the DEM for the possible locations of the robot. Geometric constraints derived from the shape and position of the HLC and the known camera geometry are used to prune the search space significantly. The search strategy is a two-stage process. Stage 1 is a coarse search that prunes the search space using geometric constraints; stage 2 refines the position using a curve matching strategy. The search algorithm presented is made robust to the errors in the various search parameters. Examples of the position estimation strategy using real terrain data are presented. Preliminary results of this work were presented in [14] and [15].

Section II briefly describes the position estimation problem of a mobile robot and summarizes the various approaches studied so far. Section III discusses the general problem of image/map correspondence and previous work in this area. Section IV describes the constrained search used in this paper and all the details of the search algorithm. Section V discusses the compuational complexity issues of the search algorithm and presents the average case, worst case, and best case order of the complexities of searching the DEM. Section VI studies the effects of the errors in the various search parameters on the position estimation and details methods to make the search algorithm robust by accounting for the errors. Section VII

Manuscript received February 20, 1991; revised December 4, 1991. This work was supported in part by the Army Research Office under Contract DAAL03-91-G-0050. Portions of this work were presented at the IEEE Workshop on Intelligent Robots and Systems (IROS '90), Japan, July 1990, and at the AIAA/NASA International Symposium on Artificial Intelligence and Robotics Applications in Space (i-SAIRAS '90), Japan, November 1990.

The authors are with the Computer and Vision Research Center, Department of ECE, University of Texas at Austin, Austin, TX 78712-1084.

IEEE Log Number 9202422.

1042-296X/92$03.00 © 1992 IEEE

presents results using real terrain data. Finally, Section VIII summarizes the work presented in this paper.

II. Position Estimation of Mobile Robots

Determining the position of the robot in its environment is one of the basic requirements for autonomous navigation. The problem of self-location has received considerable attention, and many techniques have been proposed for solving it. These techniques vary significantly depending on the kind of environment in which the robot is to navigate, the known conditions of the environment, and the type of sensors with which the robot is equipped.

In the following discussion, we consider a Cartesian coordinate system in which the *position* refers to the location of the robot (x, y) on the ground plane and the *pose* refers to the orientation of the robot, i.e., the rotation about the z axis.

Most mobile robots are equipped with wheel encoders that can be used to get an estimate of the robot's position at every instant. However, due to wheel slippage and quantization effects, these estimates of the robot's position contain errors. These errors build up and can go grow without bounds as the robot moves, and the position estimate becomes more and more uncertain. So, most mobile robots use some other form of sensing, like vision or range, to sense the environment and to aid the position estimation process.

Broadly, we can classify the position (position and pose) estimation techniques into the following types: 1) landmark-based methods; 2) methods using trajectory integration and dead reckoning; 3) methods using a standard reference pattern; and 4) methods using the *a priori* knowledge of a world model and then matching the sensor data with the world model for a position estimation.

Using landmarks for position estimation is a popular approach [10]–[13], [16]–[19]. The robot uses the knowledge of its approximate location to locate the landmarks in the environment. The landmarks can be naturally occurring in the environment, like the tops of buildings, road edges, hill tops, etc. in an outdoor scenario [17], [18] or can be identifiable beacons placed at known positions to structure the environment [12]. Once these landmarks are identified and their range/attitude relative to the robot is measured, the robot's position and pose generally can be triangulated from these measurements with a reduced uncertainty. This technique suffers from the disadvantages of requiring the availability of the landmarks in the environment and the reliance on robot's ability to detect them.

In the second type of technique, the position and pose of a mobile robot are estimated by integrating over its trajectory and dead reckoning, i.e., the robot maintains an estimate of its current location and pose at all times and, as it moves along, updates the estimate by dead reckoning [1]–[5]. However, this technique necessitates the robot's being able to establish a correspondence between the features detected by the sensors at the current location and those at the previous location (to compute the trajectory of the robot). This is, in general, not a easy problem.

The third method of accurately estimating the position and pose of the mobile robot is to place standard patterns in known locations in the environment. Once the robot images and detects these patterns, the robot's position can be estimated from the known location of the pattern and its geometry. The pattern itself is designed to yield a wealth of geometric information when transformed under the perspective projection. Different researchers [20]–[23] have used different kinds of patterns or marks, and the geometry of the method and the associated techniques for position estimation vary accordingly. These methods are only useful for indoor robots in structured environments.

In the fourth approach, the robot is aided in its navigational tasks by providing *a priori* information about the environment in the form of a preloaded world model. The basic idea is to sense the environment using on-board sensors on the robot and then to try to match these sensory observations to the preloaded world model. This process yields an estimate of the robot's position and pose with reduced uncertainty and allows the robot to perform other navigational tasks. The problem in such an approach is that the sensor readings and the world model could be in different forms. For instance, given a CAD model of a building and a visual camera, the problem is to match the 3-D descriptions in the CAD model to the 2-D visual images. This is the problem Kak *et al.* [24] address in their work on the PSEIKI system. Tsubouchi and Yuta [6] discuss the position estimation techniques used in their indoor robot, YAMABICO. The robot is equipped with a color camera and a map of the building where it is to navigate. Both are indoor robots for navigating in a building.

In this paper, we consider the position estimation problem of an autonomous land vehicle navigating in an outdoor mountainous environment. Out approach falls into the fourth type of technique outlined above. A DEM of the area in which the robot is to navigate is assumed to be given. The robot is assumed to be equipped with a camera that can be panned and tilted, an altimeter to measure the robot's altitude, and a compass. No recognizable landmarks are assumed to be present in the environment where the robot is to navigate. The DEM is a 3-D data base. It records the terrain elevations for ground positions at regularly spaced intervals. The images recorded by the camera are 2-D intensity images. The problem is to find common features to match the 2-D images to the 3-D DEM, thereby estimating the position of the robot. Since we assume the robot has a compass in this work, we only estimate the position and not the pose.

III. Image/Map Correspondence

As pointed out earlier, one of the key issues involved in determining the position of a mobile robot given a world model is to establish a correspondence between the world model (map) and the sensor data (image). Once such a correspondence is established, the position of the robot in the environment can be determined easily as a coordinate transformation. In order to solve this problem, we need to extract a set of features from the sensor data and identify the corresponding features in the world model. The problem is further complicated by the fact that the image and the map are usually in different formats.

Indeed, this problem of image/map correspondence is of fundamental importance not only to the mobile robot position estimation problem but also to many other computer vision problems in general, such as object recognition, pose estimation, airborne surveillance and reconnaissance, etc. Other work addressing this image/map correspondence problem is described in [17], [24]–[29]. The work by [27]–[29] is related more closely to our work in this paper.

Freeman and Morse [28] consider the problem of searching a *contour map* for a given terrain elevation profile. Such a problem is encountered, for example, when locating the ground track of an aircraft (the projection of the flight path on the ground) given the elevation of the terrain below the aircraft during the flight. The authors describe a solution that takes advantage of the topological properties of the contour map. A graph of the map topology is used to identify all the possible contour lines that would have been intersected by the ground track. So, the topological constraints of the terrain elevation profile and the geometric constraints of the flight path are used in estimating the location of the elevation profile in the given map. Ernst and Flinchbach [27] consider the problem of determining the correspondence between maps and the terrain images in low-altitude airborne scenarios. They assume that an initial estimate of the three-dimensional position is available. Their approach consists of partially matching the detected and expected curves in the image plane. Expected curves are generated from a map, using the estimate of the sensor position, and they match the simulated curves with the curves in the image plane. In contrast, our method does not assume the availability of an initial estimate of the sensor position. Instead we derive a set of possible positions by using a constrained search paradigm to prune the search space of possible locations. Rodriguez and Aggarwal [29] also consider the problem of matching aerial image to DEM's. They use a sequence of aerial images to perform stereo analysis on successive images and recover an elevation map. Then they present a method to match the recovered elevation map to the given DEM and thereby estimate the position and pose of the airborne sensor. All the above work, however, is concerned mainly with an airborne sensor. Our current research considers the problem of the sensor on an autonomous land vehicle. We use a constrained search strategy to isolate the robot's position.

IV. Constrained Search

One approach to this problem is to extract features from the images and then search the map for the occurrence of corresponding features. Once this correspondence is established, the position can then be computed. However, an exhaustive search of the entire map is usually prohibitively expensive if the map is very large, as in our case. As an alternative, we formulate this correspondence problem as a constrained search problem. We extract features from the images taken by the robot, and instead of searching for the corresponding features in the map, we search the space of possible robot positions for locations from where such features could be imaged. Once we isolate these sets of possible locations, then the expected features from these positions are generated; using a matching technique, the exact robot location is isolated from among these sets of possible locations. We use the horizon line as the image feature. Instead of searching using the entire feature, we use a subset of the feature and search the map for possible locations where such features can be seen. In this search strategy, the space of possible locations is first quantized to a discrete set. Then we derive geometric constraints from the image feature and the known camera geometry that need to be satisfied by any of the hypothesized positions of the robot. Using these constraints, large subspaces in this discretized search space of the possible locations are pruned as being impossible. Once the search space is pruned to a manageable size, it is searched exhaustively using the entire feature for the best estimate of the robot location.

The DEM is a 3-D data base. It is a two-dimensional array of uniformly spaced terrain elevation measurements. DEM's for various areas in the US can be obtained from the United States Geographical Survey (USGS). In this research, real terrain data were used. The position estimation algorithms developed were tested on DEM's of a 1:24 000 scale, covering different mountainous areas in Colorado and Texas. Typical DEM's were 350 by 450 samples, with a spacing of 30 m between samples. So they covered an area of typically of 140 km^2. The images recorded by the camera were 2-D intensity images. The problem was to find common features to match the 2-D images to the 3-D DEM.

The presented solution uses the DEM information and structures the problem as a constrained search in the DEM for the possible robot location. Since the robot is assumed to be located in the DEM, the DEM grid is used as a quantized version of the entire space of possible robot locations. The feature used to search the DEM is the shape and position of the horizon line contour (HLC) in the image plane. From the current robot position, images are taken in the four geographic directions: N, S, E, and W. The contour of the HLC is extracted from these images and coded. Using the height of the contour line in the image plane and the known camera geometry as input parameters, the entire DEM is searched for possible camera locations so that the points in the elevation map project onto the image plane to form a contour of the shape and height we are searching for. Since searching the DEM exhaustively for the exact shape of the horizon line is a very computationally intensive process, we split the search into two stages. In stage 1, we search using the height of the horizon line at the center of the image plane in all the four images. Geometric constraints derived from the camera geometry and the height of the HLC are used to prune large subspaces of the search space; finally, the position is isolated to a small set of possible locations. In stage 2, these locations are then considered as the candidate robot positions, and the actual image that would be seen at these points is generated using computer graphics rendering techniques from the DEM. The HLC's are also extracted from these images and then compared with the original HLC's to arrive at a measure of the new HLC's disparity. The robot location corresponding to the lowest disparity is then considered as the best estimate of the robot's position. As the results show, the approach is quite effective and, in almost all cases, the position estimate is very close to the actual position.

This approach is novel and has many advantages over previous methods. It does not rely on the existence of environmental landmarks and their detection. It does not necessitate the building of a complicated world model from the sensor observations. It is also a totally passive navigational technique since the type of sensing assumed is a single visual camera. The errors in positional estimation are not cumulative.

A. The Search Algorithm

In generating the four geographic views, the camera is assumed to have zero roll, and the optical axis of the camera is assumed to pass through the image center. The camera's tilt angle ϕ, defined as the angle between the optical axis and the horizontal plane, is adjusted until the horizon line is clearly visible in the image. This angle is then measured using an inclinometer. Commercial inclinometers that sense tilt angles up to 0.005° resolution are available. The altitude of the camera H is measured using an altimeter. This measurement is adjusted such that the altimeter reading and the height in the DEM are with respect to the same reference. Commercial barometric altimeters with a resolution of up to a few tens of feet are available. The HLC is then extracted from these images using basic image processing techniques, and the height of HLC at the center of the image plane in each of these four images is measured. Let these heights be h_i (i = N, S, E, W). The reason for using the height of the HLC at the center of the image plane is that the DEM is assumed to be gridded along N–S and E–W axes. So, the points that project onto the HLC at the center all lie along the same grid line in the DEM. Using the height of the camera H, the tilt angle ϕ_i, and the HLC height h_i in one of the directions, say north, the DEM is searched for the possible camera locations.

The algorithm *Search* is used to search the DEM in the north direction. It is similar for the other directions except that the search is carried out only in the reduced set of candidate camera positions returned by the previous search process. The direction and hence the limits of the search also vary depending on the direction in which the image is taken. It uses two pointers, one pointing to the current hypothesized camera location, CAMERA, and the other pointing to a candidate grid point, POINT. The idea is to assume that the CAMERA is at a certain point (x, y). Search along the y line to see if any POINT exists that will project at the desired height h_i onto the HLC. If so, mark this CAMERA as a possible camera position. Repeat the procedure for all the CAMERA positions along this y line and then for all y lines. The pseudo-code of the algorithm is presented below.

The Algorithm Search Stage 1 of the search process for searching in the north direction is described below.

Input: The DEM, h_N, ϕ_N, and H.

Output: A list of possible camera locations S.

```
1 Initialize the list S to zero.
2 POINT.Y = CAMERA.Y = YMIN.
   for (CAMERA.Y < YMAX) do
   3 POINT.Y = CAMERA.Y.
     POINT.X = XMIN.
     CAMERA.X = XMAX.
     for (CAMERA.X > XMIN) do
        for (POINT.X < CAMERA.X) do
        4 Z_est = Get-z-estimate(h_N, φ_N, H,
             CAMERA,POINT) /* computed
             using equation 1 */
          Z_actual = POINT.Z. /* retrieved
             from the DEM */
        5 if (Z_est ≈ Z_actual)
             Store(S,CAMERA). /* save
             current CAMERA location
             in the list*/
        6 else if(Z_actual > Z_est)
             if (Is-In-List(S,CAMERA))
                Remove(S,CAMERA).
                /* discard this CAMERA
                location if stored
                previously in the list. */
             CAMERA.X = POINT.X.
             POINT.X = XMIN. /* discard
                all the locations between
                current CAMERA and the
                POINT by updating the
                CAMERA and POINT
                locations */
        7 increment POINT.X
          od
     8 decrement CAMERA.X
       od
  9 increment CAMERA.
    od
End Algorithm Search.
```

Here, CAMERA.X and CAMERA.Y denote the x and y grid values of the location CAMERA and POINT.X and POINT.Y denote the values of the candidate point (see Fig. 1). CAMERA.X is initialized to XMAX; POINT.X to XMIN; and CAMERA.Y and POINT.Y to YMIN. Using the known approximate camera height H and the tilt angle ϕ_N, the height of the HLC, h_N, is back projected to the POINT using the geometry shown in Fig. 2. The elevation Z_{est} necessary for this POINT to project on to the HLC is estimated using (1) given below. The Appendix presents the derivation . The actual height Z_{actual} (POINT.Z) is extracted from the DEM data base. If Z_{actual} is equal to Z_{est}, then CAMERA is saved as a possible camera position. If Z_{actual} is less than Z_{est}, then the POINT is updated to a position closer to the CAMERA (in this case, POINT.X is incremented). This process is continued until POINT coincides with CAMERA. Then POINT is reinitialized, and CAMERA is updated by moving it closer to POINT (in this case, CAMERA.X is decremented). If at some stage Z_{actual} is greater than Z_{est}, then CAMERA is not a possible location since if it were, the elevation at POINT, Z_{actual}, would project a height higher than h_N. Therefore, if this CAMERA location was saved before, it is discarded. We can also say that none of the points between CAMERA and POINT are possible camera locations since at all these, the elevation at POINT, Z_{actual}, would project a height higher than h_N. This constraint proves very useful in

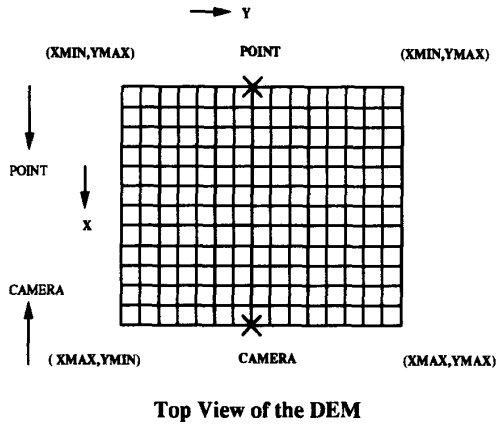

Fig. 1. The search in the north direction.

reducing the search space to a large extent for mountainous terrains with many altitude variations, such as those considered in the illustration. The search process is then repeated along all the y lines in this direction, i.e., for CAMERA.Y and POINT.Y ranging from YMIN to YMAX. The possible camera positions returned by this search process are then considered as inputs for the next search, which searches among this set with geometric constraints extracted from the image along another direction. The process is continued by applying the constraints in all the four directions, and the search refines the possible locations to a small set usually clustered around the actual location (see Fig. 14 below).

We give below an expression for the estimated elevation Z_{est} at a point in the elevation map for a hypothesized camera location and for a known imaging geometry shown in Fig. 2.

$$Z_{\text{est}} = H + x\sin\phi + x \cdot \frac{\tan\theta}{\cos\phi} \cdot \frac{h}{I_{\max}} \cdot \sqrt{\frac{h^2 + (f\phi)^2}{h^2 + f^2}} \quad (1)$$

where

$$I_{\max} = \frac{\text{image plane height}}{2}$$

f is the focal length of the camera, θ is the perspective angle, h is the HLC height at the center of the image, ϕ is the camera tilt angle, H is the altitude of the camera, and x is the distance of the point from the camera.

Another constraint used to reduce the search space is to use the approximate altitude H of the camera. Only those points of the DEM are considered as possible camera locations where the elevation Z_{actual} lies close to H, i.e., $H - \delta_H < Z_{\text{actual}} < H + \delta_H$ where δ_H is the resolution of the sensor used to measure the camera's height.

B. Curve Matching

Stage 2 of the search process further isolates the camera location from the small set of possible camera locations returned by the first stage. In this stage, each point of the set is considered as a possible candidate, and the image that would be seen in a particular direction if the camera were

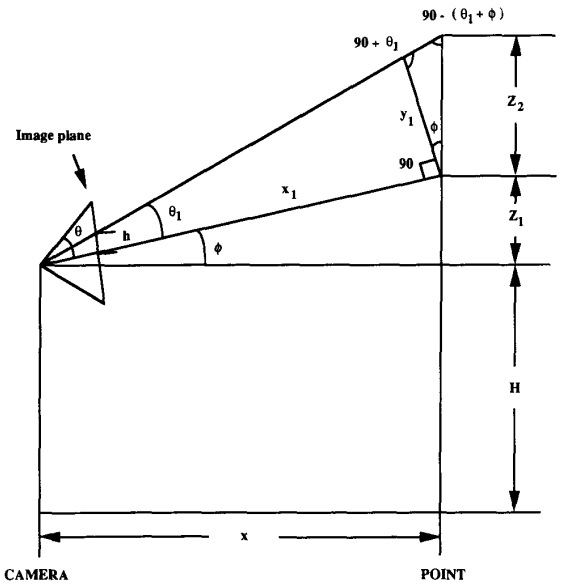

Fig. 2. The geometry of the projection.

located at that location is generated from the DEM using computer graphics rendering techniques described in Section VII. The HLC's from these images are then extracted and compared with the actual image HLC in the same direction. This is basically a 2-D curve matching problem. We have a model curve (the HLC extracted from the image) and a set of candidate curves (the HLC's generated from the candidate locations using the DEM). The objective is to find the candidate curve that matches the model curve *closest*. There is a substantial body of work in matching curves for object recognition, [26], [27], [30], and [31] to name a few. All of these mainly concern matching an image curve to a model and estimating the position and pose of the camera for the best match. Our problem is simpler than these problems because we use a search strategy to first isolate *similar* curves. Our objective is only to isolate the curve that matches to the model curve best. We use a least square technique to determine the best match. Essentially, we compute the mean square disparity between each of the candidate curves and the model curve. The candidate curve that results in the lowest mean square error is considered as the best estimate of the the robot's position.

If $m_i(i = 1, ..n)$ represents the model curve with n points and the l candidate curves are represented by $c_k(k = 1, ..l)$, each having n points $c_{ki}(i = 1, ..n)$, then for each $k(k = 1, ..l)$ we compute

$$\delta_k = \sum_{i=1}^{n} (m_i - c_{ki})^2.$$

The candidate location with the lowest δ_k is considered as the best estimate of the camera position. That means the camera location corresponding to the HLC c_k for which

$$\triangle = MIN_{k=1}^{n}\delta_k$$

is the best estimate of the camera location.

TABLE I
SEARCH RESULTS

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
(150.2,100.8)	342	7	(151,101)	(151,102)	0.824621	1.442210	2493757
(150.0,200.0)	126	3	(151,200)	(151,201)	1.000000	1.414214	2674226
(150.3,300.6)	227	8	(150,301)	(152,303)	0.500000	2.941088	2805477
(200.1,100.5)	473	8	(200,100)	(202,100)	0.509920	1.964688	2149225
(200.2,200.2)	889	26	(201,201)	(200,202)	1.131371	1.811077	2493757
(200.2,300.8)	334	9	(201,300)	(202,302)	1.131371	2.163310	3133603
(300.0,100.0)	334	4	(301,100)	(301,101)	1.000000	1.414214	2854696
(300.5,200.8)	273	17	(301,200)	(302,202)	0.943380	1.920937	1804693
(300.2,300.8)	254	10	(300,300)	(301,302)	0.824621	1.442210	2187501

(1) The actual postion.

(2) The number of possible positions after searching in one direction.

(3) The number of possible positions after searching in all directions.

(4) The estimated position (noise-free case).

(5) The estimated position (with additive Gaussian noise zero mean and standard deviation $\sigma = 5$ pixels.

(6) The error in estimated position in grid points (niose-free case).

(7) The error in estimated position in grid points (with additive noise),.

(8) The number of POINT locations explored by stage 1.

As the illustrations in Section VII show, the error measure is quite sensitive, and a location about two grid points away from the correct location results in a $\triangle$ of 48.3442. To reduce the effects of noise on the HLC, the HLC's are first smoothed using a Gaussian low pass filter. A zero mean Gaussian with a σ of 5 is used in the test runs. Due to quantization and other noise effects, this search strategy does not always isolate the position to the grid point nearest the exact location. However, the search still isolates the location to a grid point that, although not nearest the true location, is still within a small neighborhood around the true location. By using the DEM and generating the images that would be seen at different locations within a small neighborhood around the location isolated by the search, and using the curve matching strategy described in the paper, the camera location was isolated to a point nearest the true location.

This search strategy can be used in a *bootstrap* mode. That is, once the robot isolates its position in the DEM, for the subsequent navigation tasks this position estimate can be used to search only "near" the current robot location and not the entire DEM. When the location of the robot is known quite accurately, then we can in fact eliminate stage 1 and only use stage 2 of the search in a small neighborhood around the current location.

V. COMPUTATIONAL COMPLEXITY ISSUES

In this section, we discuss the computational complexity of stage 1 of the search strategy used to isolate the position of the robot in the DEM. The most computationally expensive stage of the whole search process is the search in the first direction (N, S, E, or W). After searching along one of these directions, we find that the number of possible robot locations falls to a small number, and hence searching among this reduced set using the constraints in other directions is considerably cheaper. Typically in our test runs, shown in Table I, we find that using a map of 164 063 (359 $\times$ 457) grid points, after searching in one of the directions, the number of possible robot locations falls to a few hundred locations.

Note that deriving a general formula for the exact number of computations required by the search process is not feasible, since it is completely data dependent, i.e., on the nature of the elevation values in the DEM. However, we can derive bounds on the order of complexity of the search algorithm as a function of the map size (using the probability distribution function of the elevation data in the map) and verify these with experimental results.

The algorithm *Search* described in the previous section essentially searches the entire DEM size ($M \times N$) using the HLC height h and isolates a reduced set of possible robot locations. *Search* searches along each of the N columns iteratively for the robot location. Fig. 3 shows the search along one of the columns. The x coordinate of the current camera location, CAMERA.X, runs from M to 1. For each of the CAMERA locations, we consider the POINT.X positions from 1 to (CAMERA.X - 1). And, for each set of CAMERA and POINT positions, the main computation is performed by steps 4, 5, and 6, where, for each CAMERA and POINT location, the estimated height Z_{est} is computed by back projecting h, and this is compared with the actual height at POINT given in the DEM, Z_{actual}. Depending on the result of this comparison, either the current camera location CAMERA is stored, or all the locations between CAMERA and POINT are discarded. Hence, the number of times steps 4, 5, and 6 are executed is a measure of the amount of computation performed.

A. Average Case Complexity

Consider the search along the ith column shown in Fig. 3. To determine the average amount of computation performed,

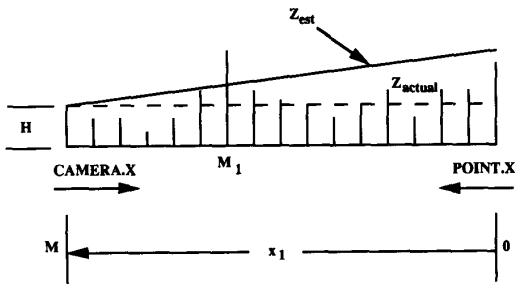

Fig. 3. The search along one of the columns.

we need to determine the grid point where the condition in Step 6, ($Z_{\text{actual}} > Z_{\text{est}}$) is satisfied in each iteration. Let x_1 be a random variable that denotes the grid point where the condition ($Z_{\text{actual}}(x_1) > Z_{\text{est}}(x_1)$) is satisfied for the first time, i.e., when CAMERA.X = M and POINT.X = 1, where $Z_{\text{actual}}(x_1)$ is the elevation in the DEM at the grid point x_1 and $Z_{\text{est}}(x_1)$ is the back-projected height at x_1 using the camera geometry, i.e., h, H, ϕ, θ.

Now the expectation of x_1 is given by

$$E(x_1) = \sum_{x_1=1}^{M-1} x_1 p(x_1)$$

where $p(x_1)$, the probability of the condition in Step 6 being satisfied at x_1, is given by

$$\begin{aligned} p(x_1) &= \text{Prob}(Z_{\text{actual}}(x_1) > Z_{\text{est}}(x_1)) \\ &= 1 - \text{Prob}(Z_{\text{actual}}(x_1) \leq Z_{\text{est}}(x_1)). \end{aligned}$$

Thus, $p(1)$, the probability of the condition in Step 6 being satisfied at $x_1 = 1$, is given by

$$\begin{aligned} p(1) &= 1 - \text{Prob}(Z_{\text{actual}}(1) \leq Z_{\text{est}}(1)) \\ &= 1 - r_1 \\ &= q_1 \end{aligned}$$

and $p(2)$ is given by the product of the probability that Step 6 is satisfied when $x_1 = 2$ and is not satisfied when $x_1 = 1$. We have

$$\begin{aligned} p(2) &= \text{Prob}(Z_{\text{actual}}(2) > Z_{\text{est}}(2)) \\ &\quad \cdot \text{Prob}(Z_{\text{actual}}(1) \leq Z_{\text{est}}(1)) \\ &= q_2 r_1 \\ p(k) &= q_k r_{k-1} r_{k-2} \cdots r_1 \\ p(M-1) &= q_{M-1} r_{M-2} \cdots r_1. \end{aligned}$$

Hence, the expectation of x_1 is given by

$$\begin{aligned} E(x_1) &= \sum_{x_1=1}^{M-1} x_1 p(x_1) \\ &= 1.q_1 + 2.q_2 r_1 + 3.q_3 r_2 r_1 \\ &\quad + \ldots + (M-1).q_{M-1} r_{M-2} \cdots r_1 \\ &= M_1. \end{aligned}$$

Thus, on the average we need to perform steps 4, 5, and 6 M_1 times before we hit the first *tall point*, where the condition in Step 6 is satisfied. In other words, we need to explore M_1 POINT positions. The CAMERA location is now updated to the grid point at $M_1 - 1$, the POINT location is reset to 1, and the search is continued. Let the location of the next grid point where the condition in Step 6 is satisfied be denoted by M_2.

Then

$$\begin{aligned} E(x_1) &= \sum_{x_1=1}^{M_1-1} x_1 p(x_1) \\ &= 1.q_1 + 2.q_2 r_1 + 3.q_3 r_2 r_1 \\ &\quad + \ldots + (M_1 - 1) q_{M_1-1} r_{M_1-2} \cdots r_1 \\ &= M_2. \end{aligned}$$

Let this search procedure continue n times before it terminates. Note that the limits of the summations are decreasing each time, i.e., $M_1 \geq M_2 \geq M_3, \ldots, M_n$. We thus have the average amount of computation performed for the ith column given by

$$C = M_1 + M_2 + M_3 \ldots M_n. \tag{2}$$

On the average, this amount of computation is to be performed for all the N columns. So, the total average computation for searching the entire map is given by NC. In the above derivation the probabilities $r_i = p(i) = 1 - \text{Prob}(Z_{\text{actual}}(i) < Z_{\text{est}}(i))$ can be computed from the probability distribution function of the DEM.

B. Upper Bound

To derive an upper bound on the computation performed by the algorithm *Search*, consider the case in which no pruning takes place in steps 4, 5, and 6 for all the rows. In this case, $M_1 = M_2 = M_3 = \cdots = M_n = M$, and $n = M$. Hence, the total number of POINT locations for which the steps 4, 5, and 6 are performed for each row is given by

$$M_1 + M_2 + \cdots + M_n = O(M^2) \quad \text{since } n = M.$$

If a similar situation occurs for all the N columns, we have the upper bound on the complexity as $O(M^2 N)$.

C. Lower Bound

To derive a lower bound, we can consider the more fortunate circumstances in which steps 4, 5, and 6 perform the maximum pruning. That is, the nature of the DEM is such that for each of the $(N-1)$ columns, which do not contain the true robot location, when CAMERA.X = M and POINT.X = 1, we find that in Step 6, the condition $Z_{\text{actual}} > Z_{\text{est}}$ is satisfied and hence all the POINT locations between POINT and CAMERA are discarded. Thus, $M_1 = 1$ and $n = 1$ and the amount of computation required is $O(1)$ for each of these $N - 1$ columns. However, for the column that contains the true robot location, the minimum computation is required if: 1) the true camera location is at CAMERA.X = M and none of the POINT.X positions from 1 to CAMERA.X satisfy the condition in Step 6. Thus, $M_1 = M$. 2) When CAMERA.X = $M - 1$ and POINT.X = 1, Step 6 is satisfied, and hence no other CAMERA location need be considered in this column. Thus, $M_2 = 1$, and hence, $n = 2$.

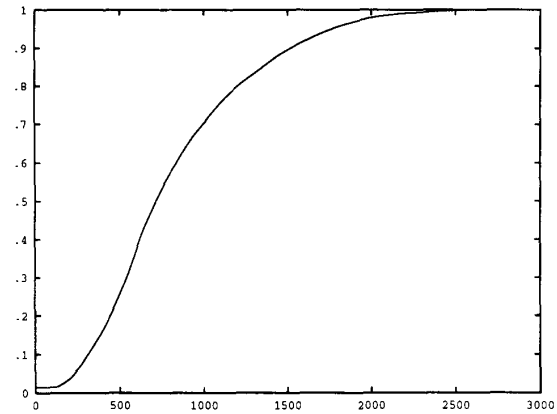

Fig. 4. The probability distribution function of elevation in the DEM.

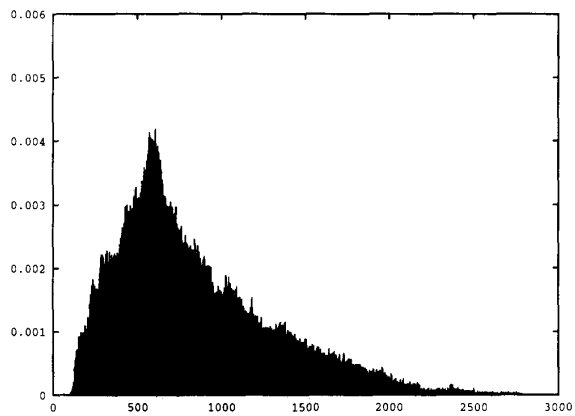

Fig. 5. The probability density function of elevation in the DEM.

The lower bound on the total amount of computation required is given by

$$(N-1)\cdot O(1)+O(M)+O(1) = O(M+N)$$
$$= O(M), \quad \text{if } M > N$$
$$= O(N), \quad \text{if } N > M.$$

Note, however, that these upper and lower bounds are quite loose. Since the data in the DEM usually represent a real mountain range, it will be much more correlated and the situations considered in deriving the bounds never happen. Also, for a given H, ϕ and θ, the the value $Z_{\text{est}}(x_1)$ depends on h, the height of the HLC. The average complexity of the search is therefore dependent on h. Fig. 6 shows a plot of the average number of POINT locations explored for different values of h, and for a given H, ϕ and θ. In this figure we used a DEM that was 359 x 457, i.e., $M = 359$ and $H = 800$ ft, $\phi = 0°$, $\theta = 33°$. The probability distribution function and the probability density function of the elevation value in the DEM used are shown in Figs. 4 and 5, respectively.

The probabilities r_i and q_i, used in computing the average amount of computation, are computed using Fig. 4. From Fig. 6 we can see that the average number of POINT locations explored grows with h. This can be explained by the fact that,

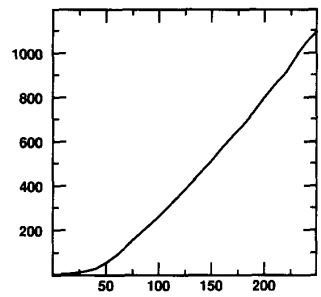

Fig. 6. The average number of POINT locations explored in each column versus h.

as h increases, for each x_1, $Z_{\text{est}}(x_1)$ increases, and hence the probability $\text{Prob}(Z_{\text{actual}}(x_1) > Z_{\text{est}}(x_1))$ decreases. Hence, we need to explore more POINT locations before the condition in Step 6 is satisfied. So, the average amount of computation performed increases with increasing h. This observation leads us to devise a search strategy in which, of the four directions N, S, E, and W that we desire to search, we search in the direction with the lowest value of h first and then the next higher h and so on. This has the effect of reducing the amount of computation required in isolating the robot location. We also observe from Fig. 6 that the average amount of computation performed is $O(M)$ for each column. For searching among all the N rows, it should be on the order of $O(MN)$.

In our experiments, in addition to computing the position we also computed the amount of computation required by counting the number of times steps 4, 5, and 6 are executed, i.e., the number of POINT locations considered for back projection. These are also shown in Table I. Note that the number of POINT locations lies between the lower bound $O(M+N)$ and the upper bound $O(M^2N)$. From the table we see that these are is in $O(MN)$ as predicted by Fig. 6. The number of POINT locations considered are actually equal to kMN, where k is a constant and $k \in (10, 20)$. Thus, the constrained search strategy serves to reduce the complexity of the search from $O(M^2N)$ in the worst case to $O(MN)$ on the average, due to the pruning.

VI. Error Analysis

The search algorithm depends upon the errors in three parameters: the error in altitude H of the camera, the HLC height h_i in the image plane, and the tilt angle of the camera ϕ_i. These three parameters affect the estimated elevation Z_{est} at a particular candidate point POINT for a given hypothesized camera position CAMERA. The errors in Z_{est} directly reflect as errors in the positional estimation, since in step 5 of the algorithm, only when Z_{est} is equal to Z_{actual} do we consider the current camera location as a possible candidate. In the rest of this section, we analyze the effects of errors in these three parameters on the computation of Z_{est} and then discuss the measures taken in the algorithm to account for the worst case case errors.

The equation relating Z_{est} to h_i, ϕ_i, and H, derived in the previous section, is given by (1) above. For a given size of the image $(2I_{\max}, 2I_{\max})$ and angle of perspective projection

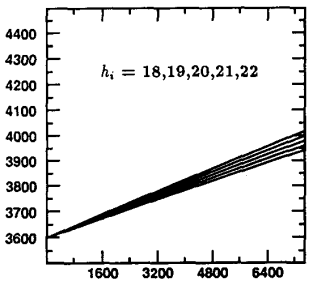

Fig. 7. Z_{est} versus x for different h_i.

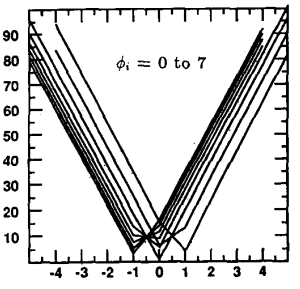

Fig. 8. ΔZ_{est} versus Δh for different ϕ_i.

θ, $\tan\theta/I_{\max}$ is a constant. Therefore, the variables are H, h_i, and ϕ_i.

Expanding the use of Taylor's series and ignoring the higher order terms, we have

$$\Delta Z_{\text{est}} \approx \Delta H \cdot \frac{\partial Z_{\text{est}}}{\partial H} + \Delta h_i \cdot \frac{\partial Z_{\text{est}}}{\partial h_i} + \Delta\phi_i \cdot \frac{\partial Z_{\text{est}}}{\partial \phi_i}. \quad (3)$$

A. Errors in H

Equation (3) shows that the error in H is directly reflected as an error in Z_{est}. To account for these, a worst case error in H is estimated from the sensor used to measure H. Let this be δH. In the search algorithm, this can be accounted for by considering the estimated elevation at the candidate point (POINT) to be acceptable if $Z_{\text{est}} - \delta H < Z_{\text{actual}} < Z_{\text{est}} + \delta H$.

B. Errors in h_i

The errors in h_i are mainly due to image quantization error and errors in the edge detector used to extract the horizon line. Since x, the distance between POINT and CAMERA, occurs as a multiplicative parameter in the second and third terms of (1), the errors in h_i are magnified by this and, hence, vary depending on x. The worst error compensation should take this into account. Fig. 7 shows a typical plot of Z_{est} versus x for different Δh_i. As can be seen, with increasing distance the error in Z_{est} grows almost linearly. Fig. 8 shows a plot of ΔZ_{est} versus Δh_i for different ϕ_is and for fixed x and H. Ideally, $\Delta Z_{\text{est}} = 0$ should occur at $\Delta h_i = 0$. However, due to image quantization errors, this occurs for $\Delta h_i \in (-1, +1)$, as can be seen for different values of ϕ_i. One way to account for these errors in h_i is to back-project a band of h_i values, $(h_i - \Delta h_i, h_i + \Delta h_i)$, instead of a single h_i in the estimation of Z_{est}. Thus, for each CAMERA and POINT location, we obtain a range of acceptable elevations $(Z_{\text{est}} + \Delta Z_{\text{est}}, Z_{\text{est}} - \Delta Z_{\text{est}})$ and we consider CAMERA as a possible camera location only if Z_{actual} at this POINT lies within this range.

C. Errors in ϕ_i

Similar to h_i, the effects of errors in ϕ_i are also magnified by x and, hence, depend on the distance between CAMERA and POINT. The algorithm is very sensitive to errors in ϕ_i. As Fig. 9 shows, an error of 0.5° in ϕ_i causes an error of about 50 m in Z_{est} for the given x, H, and h_i. Therefore, ϕ_i has to be measured accurately or at least represented by a good worst case estimate so that, as before, Z_{est} can be compensated for.

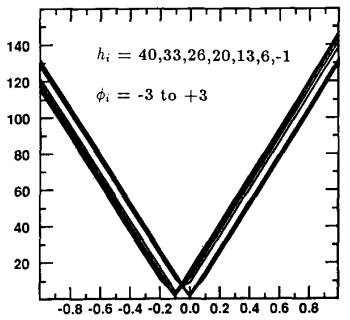

Fig. 9. ΔZ_{est} versus $\Delta\phi$ for different h_i.

In the actual implementation of the algorithm, errors in ϕ_i are accounted for by considering the effect of these errors as errors in h_i. That is, the band $h_i \pm \Delta h_i$ is made wider to account for $\Delta\phi_i$. Hence, the range of acceptable Z_{est} values is increased. In the illustrations considered, we find that a $\Delta h_i = 3$ pixels accounts for a 0.5° error in ϕ_i and for a ±1 pixel error in h_i.

Another observation from Fig. 9 is that $\Delta Z_{\text{est}} = 0$ does not always occur at $\Delta\phi_i = 0$. This is due to the roundoff errors in the computation of Z_{est} from ϕ_i, H, h_i, and x, which is a reasonably complicated trigonometric expression.

VII. Illustrations

The algorithms developed in this research have been tested using real terrain data obtained from the USGS of various areas in Colorado and Texas using synthetic images generated from the DEM. This section details the results of a typical run using a DEM of an area in Colorado. The elevation data is a uniform square grid of 30-m resolution and has 359 × 457 grid points. It covers an area of 148 km^2. Synthetic images for an assumed camera location are generated from the DEM using the AT&T Pixel machine, to serve as the images taken by the robot. The elevation data are tessellated into polygons, and surface normals are calculated at each of the elevation points using the four neighboring points. A light source position and direction are assumed, and, for a given camera location and perspective geometry, Gouraud-shaded polygons are drawn and projected onto the image plane to generate perspective views of the mountain range.

Figs. 10–16 illustrate a typical run of the algorithm. Fig. 10 shows a typical image used to test the algorithm. Fig. 11 shows the HLC extracted from this image using a gradient

Fig. 10. A typical view.

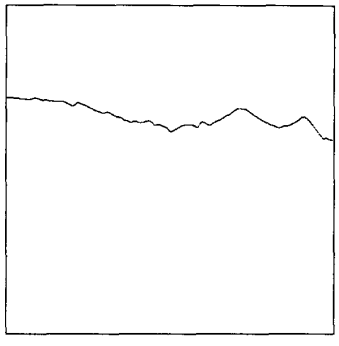

Fig. 11. The extracted HLC.

Fig. 12. A top view of the environment of the robot.

Fig. 13. The possible camera locations after searching in one direction.

operator. Fig. 12 shows a top view of the environment of the robot. In this example, of the possible 164 063 (359 $\times$ 457) possible locations, stage 1 of the search process using h_N and the associated camera geometry returns 473 points as possible camera locations. Fig. 13 shows a top view of these locations. These successively drop in the consecutive searches that use h_i in the other directions. Ultimately, we have about eight possible camera locations. Fig. 14 shows a top view of these locations. Figs. 15 and 16 show the results of the second stage of the search process. The figures show the HLC's of some of the images generated in the north direction for the candidate camera locations returned by the first stage of the search process. The figures also show a measure of the difference between these and the actual image HLC in this direction. The measure used is a mean square error between the two contours. The candidate with the lowest error, in this case 48.3442 units, is then considered as the estimate of the camera location. The actual position is (200.1,100.5) and the estimated location is (201,102)). The four neighbors of this point are then considered as candidates; images are also generated in the north direction for these positions. The HLC's are then extracted and compared against the original HLC, and the best estimate of the location is isolated as the one with the least mean square error.

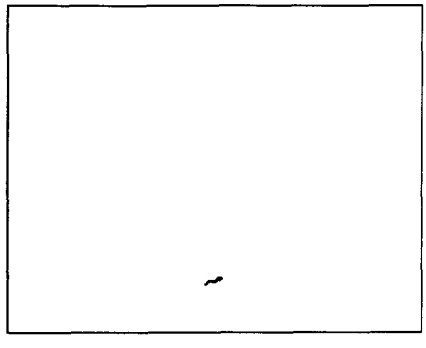

Fig. 14. The possible camera locations after searching in all four directions.

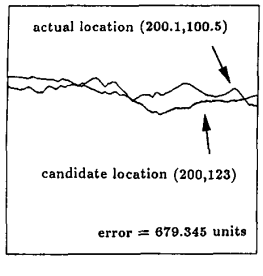

Fig. 15. The disparity between HLC's.

The results of running the algorithm at various locations on the DEM are shown in Table I. As can be seen, in most cases the estimated position is quite close to the actual location. We then add zero-mean Gaussian noise, with a standard deviation of 5, to the horizon lines. This represents the noise in the image formation process and the noise in the detection of the HLC.

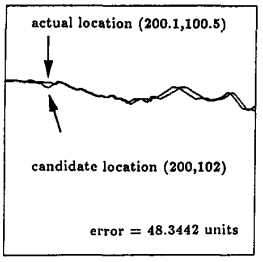

Fig. 16. The disparity between HLC's.

Since the HLC's are smoothed by a Gaussian filter and since we actually use a band of h_i values in the back projection, the effects of the additive Gaussian noise are not felt much. As the test runs illustrate, the effect of this noise is to shift slightly the estimated position in some cases. However, the estimated position is still quite close to the actual location in all cases. So, the method is quite robust to errors in the imaging process. From the table, the average error in the estimated location over the nine test runs is computed to be 0.8739 grid points in the noise free case and to be 1.83488 in the case with additive Gaussian noise (each grid point corresponds to 30 m in the DEM). The amount of computation performed by stage 1 is also shown in this table as the number of POINT locations explored by the search. We note that this is in $O(MN)$.

VIII. Conclusions

This paper presents a novel method for mobile robot localization in an outdoor mountainous environment given a DEM of the region. It is not assumed that the robot is capable of recognizing any landmarks. The height and shape of the horizon line in the image plane is used as the main feature, to search the DEM for the possible robot location. A constrained search paradigm is used to reduce the search space and, hence, to speed up the search process. The effect of errors in the various parameters on the positional estimation is studied and the algorithm is made robust by taking into account the worst case errors. The algorithm is tested on the DEM's of different mountainous locations in Colorado and Texas using images simulated from the DEM with good success in isolating the position.

In a real-world application it is possible that the HLC's in some of the four principal directions may be occluded; in these cases, one possible approach is to interpolate the DEM and resample it in different directions in which the HLC is more clearly visible. The method is found to be sensitive to errors in the tilt angle ϕ_i of the camera. However, it is possible to measure this angle quite accurately by using a precision inclinometer. Also, by back projecting a larger band of around the HLC height h, as discussed in Section VI, it is possible to account for the errors in the input parameters. Sometimes it may so happen that the robot's location cannot be unambiguously established to one location in the DEM but only to a set of possible locations. One way to disambiguate between all these possible locations is to translate the robot by a known amount in a known direction, image the environment, and apply the search strategy again with additional constraints imposed by these new measurements.

Since the method relies on the uniqueness of the horizon line, the approach is limited to use in areas of reasonable altitude variations.

Appendix
The Derivation of Z_{est}

Here, we derive an expression for Z_{est} from the geometry of Fig. 2 using the following variables:

$$I_{\max} = \frac{\text{image plane height}}{2}$$

f is the focal length of the camera, θ is the perspective angle, h is the HLC height at the center of the image, ϕ is the camera tilt angle, H is the altitude of the camera, and x is the distance between CAMERA and POINT. The derivation is as follows:

$$Z_{\text{est}} = H + Z_1 + Z_2 \tag{A1}$$

$$Z_1 = x \sin\phi \tag{A2}$$

$$\tan\theta = \frac{I_{\max}}{f}$$

$$\tan\theta_1 = \frac{h}{I_{\max}} \cdot \tan\theta$$

$$x_1 = \frac{x}{\cos\phi}$$

$$y_1 = x_1 \cdot \tan\theta_1 = \frac{x}{\cos\phi} \cdot \frac{h}{I_{\max}} \cdot \tan\theta$$

$$Z_2 = y_1 \cdot \frac{\sin(90+\theta_1)}{\sin(90-(\theta_1+\phi))} = y_1 \cdot \frac{\cos(\theta_1)}{\cos(\theta_1+\phi)}$$

$$= \frac{x}{\cos\phi} \cdot \frac{h}{I_{\max}} \cdot \tan\theta \cdot \frac{\cos\left[\tan^{-1}\left(\frac{h}{I_{\max}} \cdot \tan\theta\right)\right]}{\cos\left[\tan^{-1}\left(\frac{h}{I_{\max}} \cdot \tan\theta + \phi\right)\right]}. \tag{A3}$$

Finally from (A1), (A2), and (A3), we have

$$Z_{\text{est}} = H + x\sin\phi + x \cdot \frac{\tan\theta}{\cos\phi} \cdot \frac{h}{I_{\max}} \cdot \frac{\cos\left[\tan^{-1}\left(\frac{h}{I_{\max}} \cdot \tan\theta\right)\right]}{\cos\left[\tan^{-1}\left(\frac{h}{I_{\max}} \cdot \tan\theta + \phi\right)\right]}. \tag{A4}$$

Substituting $\tan\theta = I_{\max}/f$ and simplifying, we have

$$Z_{\text{est}} = H + x\sin\phi + x \cdot \frac{\tan\theta}{\cos\phi} \cdot \frac{h}{I_{\max}} \cdot \sqrt{\frac{h^2 + (f\phi)^2}{h^2 + f^2}}. \tag{A5}$$

References

[1] R. Chatila and J.-P. Laumond, "Position referencing and consistent world modeling for mobile robots," in *Proc. IEEE Int. Conf. Robotics Automat.* (St. Louis, MO), Mar. 1985, pp. 138–145.

[2] J. L. Crowley, "Dynamic world modeling for an intelligent mobile robot using a rotating ultra-sonic ranging sensor," in *Proc. IEEE Int. Conf. Robotics Automat.* (St. Louis, MO), Mar. 1985, pp. 128–135.

[3] O. Faugeras and N. Ayache, "Building, registering and fusing noisy visual maps," in *Proc. 1st Int. Conf. Computer Vision* (London), June 1987, pp. 73–82.
[4] L. Matthies and S. A. Shafer, "Error modeling in stereo navigation," *IEEE J. Robotics Automat.*, vol. RA-3, pp. 234–248, June 1987.
[5] H. P. Moravec, *Robot Rover Visual Navigation*. Ann Arbor, MI: UMI Research Press, 1981.
[6] T. Tsubouchi and S. Yuta, "Map assisted vision system of mobile robots for reckoning in a building environment," in *Proc. IEEE Int. Conf. Robotics Automat.* (Raleigh, NC), Mar. 1987, pp. 1978–1984.
[7] D. M. Keirsey, D. W. Payton, and J. K. Rosenblatt, "Autonomous navigation in cross country terrain," in *Proc. Image Understanding Workshop, DARPA* (Cambridge, MA), Apr. 1988, pp. 411–416.
[8] C. Thorpe and T. Kanade, "Carnegie Mellon Navlab vision," in *Proc. Image Understanding Workshop, DARPA* (Palo Alto, CA), May, 1989, pp. 273–282.
[9] C. Fennema and A. R. Hanson, "Experiments in autonomous navigation," in *Proc. 10th Int. Conf. Pattern Recognition* (Atlantic City, NJ), 1990, pp. 24–31.
[10] B. C. Bloom, "Use of landmarks for mobile robot navigation," in *Proc. SPIE, Intell. Robots Comput. Vision*, vol. 579, pp. 351–355, 1985.
[11] M. Case, "Single landmark navigation by mobile robots," in *Proc. SPIE, Mobile Robots*, vol. 727, pp. 231–238, Oct. 1986.
[12] C. D. McGillem and T. S. Rappaport, "Infra-red location system for navigation of autonomous vehicles," in *Proc. IEEE Int. Conf. Robotics Automat.* (Philadelphia, PA), Apr. 1988, pp. 1236–1238.
[13] H. Nasr and B. Bhanu, "Landmark recognition for autonomous mobile robots," in *Proc. IEEE Int. Conf. Robotics Automat.*, 1988, pp. 1218–1223.
[14] R. Talluri and J. K. Aggarwal, "Positional estimation for a mobile robot in an outdoor environment," in *Proc. IEEE Workshop Intell. Robots Syst., IROS '90* (Japan), July 1990, pp. 159–166.
[15] R. Talluri and J. K. Aggarwal, "A positional estimation technique for an autonomous land vehicle in an unstructured environment," in *Proc. AIAA/NASA Int. Symp. Artificial Intell. Robotics Applications in Space, i-SAIRAS '90* (Kobe, Japan), Nov. 1990, pp. 135–138.
[16] E. Krotkov, "Mobile robot localization using a single image," in *Proc. IEEE Int. Conf. Robotics Automat.* (Scottsdale, AZ), May 1989, pp. 978–983.
[17] T. S. Levitt, D. T. Lawton, D. M. Chelberg, and P. C. Nelson, "Qualitative navigation," in *Proc. Image Understanding Workshop, DARPA* (Los Angeles), Feb. 1987, pp. 447–465.
[18] R. Kumar, "Determination of the camera location and orientation," in *Proc. Image Understanding Workshop, DARPA* (Cambridge, MA), Apr. 1988, pp. 870–881.
[19] K. Sugihara, "Some location problems for robot navigation using a single camera," *Comput. Vision, Graphics and Image Processing*, vol. 42, pp. 112–129, Apr. 1988.
[20] I. Fukui, "TV image processing to determine the position of a robot vehicle," *Pattern Recognition*, vol. 14, no. 1-6, pp. 101–109, 1981.
[21] J. Courtney, M. Magee, and J. K. Aggarwal, "Robot guidance using computer vision," *Pattern Recognition*, vol. 17, no. 6, pp. 585–592, 1984.
[22] M. J. Magee and J. K. Aggarwal, "Determining the position of a robot using a single calibration object," in *Proc. IEEE Int. Conf. Robotics* (Atlanta), Mar. 1984, pp. 140–149.
[23] M. R. Kabuk and A. E. Arenas, "Position verification of a mobile robot using a standard pattern," *J. Robotics Automat.*, vol. RA-3, pp. 505–516, Dec. 1987.
[24] A. C. Kak, K. M. Andress, C. Lopez-Abadia, and M. S. Carroll, "Hierarchical evidence accumulation in the PSEIKI system and experiments in model-driven mobile robot navigation," in *Uncertainty in Artificial Intelligence*, vol. 5. Amsterdam, The Netherlands: Elsevier, pp. 353–369, 1990.
[25] G. Medioni and R. Nevatia, "Matching images using linear features," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, pp. 675–685, Nov. 1984.
[26] F. Mokhtarian and A. Mackworth, "Scale-based description and recognition of planar curves and two-dimensional shapes," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-8, pp. 34–43, Jan. 1986.
[27] M. D. Ernst and B. E. Flinchbaugh, "Image/map correspondence using curve matching," Texas Instruments, Tech. Rep. CSC-SIUL-89-12, 1989.
[28] H. Freeman and S. P. Morse, "On searching a contour map for a given terrain elevation profile," *J. Franklin Inst.*, vol. 284, pp. 1–25, July 1967.
[29] J. J. Rodriguez and J. K. Aggarwal, "Matching aerial images to 3-d terrain maps," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, pp. 1138–1149, Dec. 1990.
[30] J. T. Schwartz and M. Sharir, "Identification of partially obscured objects in two and three dimensions by matching noisy characteristic curves," *Int. J. Robotics Res.*, vol. 6, no. 2, pp. 29–44, 1987.
[31] H. Wolfson, "On curve matching," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, pp. 483–489, May 1990.

Raj Talluri received Bachelor of Engineering degree in electronics and Communications Engineering from Andhra University, India, in 1984 and the Master of Engineering degree in applied electronics from Anna University, India, in 1986. He is currently working toward the Ph.D. degree at the Computer and Vision Research Center, Department of Electrical and Computer Engineering, University of Texas at Austin.

He has been a Graduate Research Assistant at the Computer and Vision Research Center since June 1988. He worked as a scientist at the Indian Space Research Organisation's Satellite Center, Bangalore, India, from 1986 to 1987. He was a Graduate Research Assistant in the bio-chemistry program at the University of Texas at Austin from January to August 1988. He has also worked as a student member of the corporate technical staff at the Central Research Laboratories, Texas Instruments, Dallas, from May to August 1991. His research interests include computer vision, autonomous navigation, robotics, and computer graphics.

Mr. Talluri is a student member of the IEEE Computer Society.

J. K. Aggarwal (S'62–M'65–SM'74–F'76) has served on the faculty of the University of Texas at Austin in the College of Engineering since 1964 and is now the Cullen Professor of Electrical and Computer Engineering and Director of the Computer and Vision Research Center. His research fields incolude computer vision, parallel processing of images, and pattern recognition. He is the author or editor of 6 books and 20 book chapters and the author of over 130 journal papers, as well as numerous proceedings papers and technical reports.

Dr. Aggarwal has been a Fellow of the IEEE since 1976. He served as Chairman of the IEEE Computer Society Technical Committee on Pattern Analysis and Machine Intelligence (1987–1989), was Director of the NATO Advanced Research Workshop on Multisensor Fusion for Computer Vision (Grenoble, France, 1989), and was Chairman of the Computer Vision Program Committee of the 10th International Conference on Pattern Recognition (Atlantic City, NJ, 1990). He received the 1992 Senior Award of the American Society of Engineering Education. He is the IEEE Computer Society representative to the International Association for Pattern Recognition, Treasurer of the International Association for Pattern Recognition, and Editor of IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS.