Mobile Robot Self-Location Using Model-Image Feature Correspondence

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

Abstract— **The problem of establishing reliable and accurate correspondence between a stored 3-D model and a 2-D image of it is important in many computer vision tasks, including model-based object recognition, autonomous navigation, pose estimation, airborne surveillance, and reconnaissance. This paper presents an approach to solving this problem in the context of autonomous navigation of a mobile robot in an outdoor urban, man-made environment. The robot's environment is assumed consist of polyhedral buildings. The 3-D descriptions of the lines constituting the buildings' rooftops is assumed to be given as the world model. The robot's position and pose are estimated by establishing correspondence between the straight line features extracted from the images acquired by the robot and the model features. The correspondence problem is formulated as a two-stage constrained search problem. Geometric visibility constraints are used to reduce the search space of possible model-image feature correspondences. Techniques for effectively deriving and capturing these visibility constraints from the given world model are presented. The position estimation technique presented is shown to be robust and accurate even in the presence of errors in the feature detection, incomplete model description, and occlusions. Experimental results of testing this approach using a model of an airport scene are presented.**

I. Introduction

COMPUTER vision researchers are interested in building intelligent robots that can navigate autonomously without human intervention. The uses of such an autonomous mobile robot are numerous, ranging from providing access to hazardous industrial environments to battlefield surveillance vehicles. However, a number of issues and problems must be addressed in the design of an autonomous mobile robot, from basic scientific issues to state-of-the-art engineering techniques [1].

A mobile robot can be assisted in its navigation tasks by providing it with *a priori* knowledge about the environment in which it will navigate, usually called a *world model* or a *map*. This map can be in various forms, such as a CAD description of the environment, a floor map, or, in the case of an outdoor terrain robot, a Digital Elevation Map (DEM). One of the issues to be addressed in using a stored model as an aid in mobile robot navigation is that of estimating the position and orientation of the robot with respect to the model. Once the robot accurately estimates its location within the model, other navigation tasks can be performed. Most mobile robots are equipped with wheel encoders that can estimate the robot's relative position at every instant; however, due to wheel slippage and quantization effects, these estimates of the robot's position contain errors. These errors accrue and can grow limitlessly as the robot moves, causing the position estimate to become increasingly uncertain. So, most mobile robots use additional forms of sensing, such as vision to aid the position estimation process. Talluri and Aggarwal [2] present a comprehensive review of the different position estimation techniques for mobile robot self-location.

In order to effectively use the stored world model of the environment and the sensor data, it is necessary to establish correspondence between the sensory observations and the model information. Typically a visual sensor such as a CCD camera is used to record an optical intensity image of the scene. Depending on the environment, the model may be a floor map, a CAD model (in the case of indoor robots), or a Digital Elevation Model (DEM) (in the case of outdoor robots). The problem of model-image correspondence is particularly hard since usually the model and image are in different formats, described in different coordinate frames and of different dimensions.

This model-image correspondence problem is important in many other computer vision tasks, such as model-based object recognition, pose estimation, airborne surveillance, and reconnaissance. A popular approach to solving this problem is to extract features from the image and search for the corresponding set of features in the model description. The type of features required and the number of features used depends on the model description and what is assumed to be known about the scene. For example, in navigating a robot in an indoor structured environment with a given CAD model of the environment, it is common practice to use line segments as features [3]. On the other hand, in navigating a robot in an outdoor mountainous terrain given a DEM of the environment, using curves may be a logical choice [4]. Typically, in these problems the model and the camera (robot) use two different coordinate systems. Once we extract the relevant features from the image and identify the corresponding features in the model, we can compute the transformation that maps the model features into the image features. The parameters of this transformation are the required position and pose of the camera (robot) with respect to the model. Solving for these parameters, once a set of model-image feature correspondences is established, is a very well studied problem [5].

In the context mobile robot self-location, it is possible to exploit some of the constraints imposed by the fact that

Manuscript received April 16, 1993; revised September 14, 1994.

R. Talluri is with the Systems and Information Science Laboratory, Corporate Research, Texas Instruments, Dallas, TX 75265 USA.

J. K. Aggarwal is with the Computer and Vision Research Center, Department of ECE, University of Texas, Austin, TX 78712 USA.

Publisher Item Identifier S 1042-296X(96)00968-2.

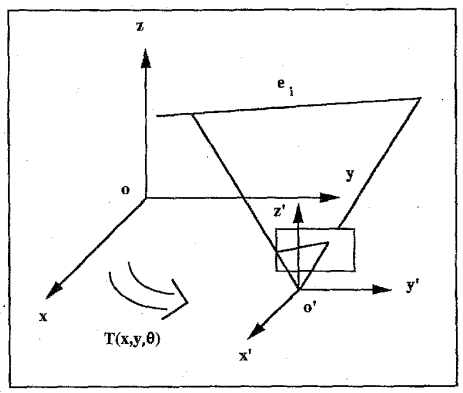


Fig. 1. The world and robot coordinate systems.

most mobile robots navigate on the ground plane in reducing the number of degrees of freedom in the transformation that maps the world model features into the image features. Consider the world coordinate system $OXYZ$ and the robot coordinate system $O'X'Y'Z'$ shown in Fig. 1. Generally, the transformation $\mathcal{T}$ that transforms $OXYZ$ into $O'X'Y'Z'$ has six degrees of freedom: three rotational and three translational. Most mobile robot self-location tasks make the assumption that the robot is on the ground (OXY plane), so the Z-translation (the height of the robot above the ground) is assumed to be known or to be zero. The camera on the robot is assumed to have zero roll (rotation about X-axis), and the tilt angle of the camera, (rotation about the Y-axis) is assumed to be measurable. So, there are effectively three parameters in the transformation: two translational (X, Y) and one rotational θ (the pan angle of the camera, which is a rotation about the Z-axis). The parameter space of the transformation is thus the entire OXY plane and the range of robot orientation θ is 0–360°. In the rest of this paper, *position* refers to the robot's location on the ground plane and *pose* refers to the robot's orientation We use the term *position estimation* to refer to both position and pose estimation.

Therefore, the crucial task to be accomplished is that of establishing a reliable and accurate correspondence. Noise, occlusions, errors in feature detection, and inaccurate model descriptions further complicate this correspondence problem. In this research we consider this model-image correspondence problem and its application to the task of mobile robot self-location. The robot is assumed to be navigating in an outdoor urban, man-made environment. The environment of the robot is assumed to consist of polyhedral buildings with *flat* (parallel to the ground plane) rooftops. The 3-D descriptions of the line segments composing the buildings' rooftops make up the world model. It is possible to obtain such a description from a CAD description of the environment. The robot images the environment using a single CCD camera. The position and pose of the robot are estimated by establishing correspondence between the world model features (the buildings' rooftops) and the image features (lines extracted from the image). We present a two-stage constrained-search strategy that draws from the existing body of work in model-image feature correspondence to establish a reliable and accurate correspondence even in the presence of errors in feature detection, incomplete world model descriptions, and occlusions.

We argue for the use of precompiled recognition strategies and the idea that the complexity of the search lies in the world model rather than the image. We wish to exploit the geometric visibility constraints imposed by the world model description in pruning the search space of all possible model-image feature correspondences. In order to capture these constraints effectively, we propose a new representation of the environment known as the *Edge Visibility Regions* (EVR's). We present an algorithm to construct such an EVR description of the environment from the given world model description. The EVR description is a partitioning of the free space of the robot into distinct, nonoverlapping regions. Each region has an associated *Visibility List* (VL) of the world model features visible in that region. The EVR's are essentially a mechanism to group the model features based on geometric visibility constraints.

Once an EVR description of the environment is formed off-line from the given world model, a two-stage constrained-search strategy isolates the robot's location using the EVR description. The first stage is a modified Hough transform technique that uses the image features, the EVR description, and a transform clustering strategy to narrow the robot's location to a small set of possible regions. This modified Hough transform technique avoids the shortcomings of the traditional Hough transform approaches [6] but retains the noise immunity provided by transform clustering.

The second stage of the search is a fine search among the reduced set of possible locations returned by the first stage to establish an accurate set of model-image feature correspondences and estimate the robot's position and pose in its environment. We use an interpretation-tree search technique to establish model-image feature correspondences. The geometric constraints captured by the EVR's are used to prune large subspaces in the tree. Also, since the EVR's provide an effective mechanism to capture the visibility constraints based on the 3-D model geometry, we are able to extend the traditional interpretation-tree and constrained-search approaches [7], [8] to situations where the model data is 3-D and the image data is 2-D. One of the drawbacks of the constraints used by Grimson is that the model and sensor data need to be of the same dimensionality, i.e., 2-D data and 2-D models or 3-D data and 3-D models. His constraints cannot be extended to the case of a 3-D model and a 2-D image since the camera's perspective projection geometry destroys these length, angle and distance constraints.

Unlike previous approaches [9], [10] that use the weak perspective approximation, we handle the model-image feature correspondence under full perspective. Also, unlike previous research [3], in this research we solve the *drop-off* problem [11], where the robot does not have a good initial estimate of its location. However, we show that when the robot has a good initial estimate of its position, this information can be used to significantly reduce the search complexity. In this research we describe model-image feature correspondence strategies that exploit the advantages of grouping, transform clustering, and constrained search while avoiding some of their drawbacks.

The effectiveness of the technique is demonstrated by applying it to the problem of mobile robot self-location. Experimental results demonstrating the effectiveness of this approach using a model of an airport scene are also presented.

The rest of the paper is organized as follows. Section II presents an algorithm for deriving a new representation of the environment of the mobile robot from the given stored world model. This representation implicitly captures the geometric visibility constraints established the world model. Section III details a two-stage constrained-search strategy that uses these constraints to establish an accurate and reliable correspondence between the model and the image and, hence, estimate the position and orientation of the robot in its environment. Section IV presents the experimental results of testing the ideas presented in this paper using a model of an airport scene. Finally, Section V summarizes the ideas presented in this research and points out future research directions.

II. Edge Visibility Regions

This section presents an algorithm for capturing the geometric visibility constraints imposed by the robot's world model description [12]. The constraints are captured implicitly by developing a new representation of the environment knows as Edge Visibility Regions (EVR's) from the given stored model. The EVR's essentially provide a mechanism for grouping the model features based on their visibility from different positions on the ground.

We assume that the robot's environment consists of polyhedral buildings, and that the 3-D descriptions of the buildings' rooftops is given as the world model. Initially, we consider the restrictive case that all the buildings have *flat* (parallel to the ground plane), convex rooftops and are the same height. We then develop an algorithm to generate the EVR's and their associated VL's. The extension of this algorithm to the general case of buildings of unequal heights and nonconvex rooftops is then discussed. In the former case, only the projections of the buildings onto the xy-plane need to be considered in forming the EVR's and VL's, since the tilt angle ϕ is assumed to be measurable.

One problem similar to this problem is that of forming the aspect graph of a polyhedral object for object recognition [13]–[19]. The aspect graph provides a complete enumeration of all possible distinct views of an object, given a model for viewpoint space and a definition for *distinct*. Bowyer and Dyer [19] provide a good summary of existing aspect graph techniques for object recognition. However, the existing aspect graph methods concern mainly a single polyhedral object and divide the entire 3-D view space into regions based on the visibility of the object's faces, edges, and vertices. As a result, although the aspect graph representation is an interesting idea for use in object recognition, the representations are usually very complex and limited to single, convex polyhedra. So far no existing, working systems can handle cases of multiple, general polyhedra [20].

The present work tries to capture the essence of the aspect graph representation by forming a partitioning of the robot's free space into the EVR's. However, we alleviate the problems of the traditional aspect graph representations by considering only a small subset of the model features–the rooftop lines. Furthermore, since we are only concerned with estimating the position and pose of an autonomous mobile robot, (x, y, θ), we need to consider only one plane, the ground plane, as the viewing space. These constraints reduce the complexity of the representation significantly while exploiting its advantages. Also, we can now handle multiple, nonconvex buildings in arbitrary orientations.

Hence, our approach is a simplification of the aspect graph representation, to the specific case of the camera being located on a single plane. However, our approach does consider the occlusions and the various other visual events that arise due multiple polyhedral objects and occlusions between them. In view of these differences, we found that the existing aspect graph algorithms could not be extended to handle the this specific problem. We thus derive a new technique for carving the free space into visibility regions using which is similar in spirit to the aspect graph algorithms but exploits the simplifications offered by the constrained viewpoint.

There is a significant body of work in computational geometry on the various visibility problems that arise while considering polygons [21]–[23]. Toussaint [22] provides a comprehensive review of the visibility properties of polygons. However, although our work has some similarities to some of the problems studied, none of them are directly applicable since we deal with the visibility properties of multiple polyhedral objects and under perspective projection. This 3-D nature of the world model and the projective geometry of the camera inhibit the direct applicability of the existing computational geometry techniques for polygon visibility. Nonetheless, we do exploit some of the techniques from computational geometry in computing the EVR's from the given world model description.

We describe below the algorithm `Partition` that divides the xy plane into the desired EVR's, along with their associated VL's. The algorithm uses three subprocesses called `Split`, `Project`, and `Merge`. The algorithm's basic idea is to start with the entire xy-plane as one EVR with a NULL visibility list. Each polygon is considered in turn by extending each of its edges, and the EVR's that are intersected are divided into two new ones. The new EVR's then replace the old one, and the VL's of the new EVR's are updated to account for the visibility of this edge by considering it to be visible in one half-plane, say the half-plane into the left of the edge, and invisible in the other. `Split` process handles this updating. For each new polygon considered, the mutual occlusion of the polygon's edges with the other existing polygons is handled by forming the *shadow region* of these edges on the other existing polygons. `Project` handles the forming of these shadow regions. Finally, the `Merge` process concatenates all the adjacent EVR's with identical VL's into one EVR.

After partitioning the xy-plane into EVR's, the range of the robot's orientations for which each model edge in the VL of an EVR is visible is also computed and stored. An efficient method to compute these ranges is also described.

`Split`: Given a 2-D convex polygon in a plane and a set of existing EVR's, the `Split` process updates the

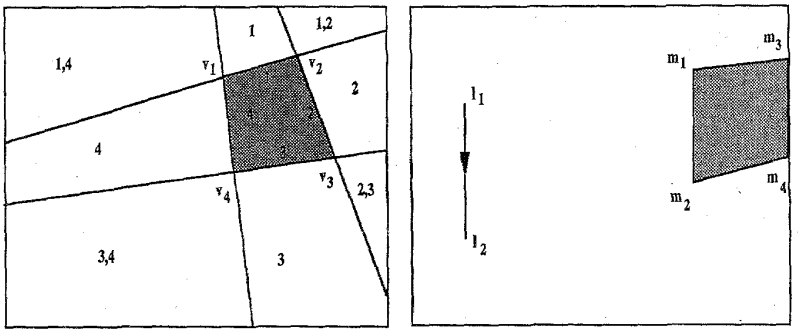


Fig. 2. (a) The EVR's after `Split`. (b) The shadow region for two lines.

EVR list to account for the visibility of the polygon's edges. Consider an n-sided convex polygon P in the viewing plane defined as a collection of n vertices, $v_1, v_2, \cdots v_n$, and n edges, $v_1v_2, v_2v_3, \cdots v_{n-1}v_n, v_nv_1$, such that no pair of nonconsecutive edges shares a point. Then

1) extend each of the edges of P, $v_1v_2, \cdots, v_nv_1$, in order and for each of these lines; and
2) check if this line cuts an existing EVR in the current EVR list. If it does, split the EVR into two along this line, copy the VL of the EVR into the two new EVR's, update the EVR list by replacing the old EVR by the two newly formed ones, and update the VL of the region to the left of this line by adding this edge, v_iv_{i+1}, to its VL.

Fig. 2 shows the EVR's and their VL's after extending all the edges of P.

`Project`: The `Project` is used to account for the mutual occlusions between lines and polygons that affect their visibility from different locations in the viewing plane. The process is first described for two line segments and then for a line segment and a polygon.

Case 1: Consider the case of two lines, l_1l_2 and m_1m_2, in a plane, as Fig. 2(b) shows. We now need to find the region in the plane where l_1l_2 is not visible due to occlusion from m_1m_2. Note that m_1m_2 lies to the left of (the visible side of) the directed line segment l_1l_2. Let us refer to this region as the *shadow region* of m_1m_2 due to l_1l_2. One way to find this shadow region is to extend the line l_1m_1 and then to find the intersection of this line with the viewing plane, as m_4; similarly, extend l_2m_2 and find m_3. Now the region $m_1m_2m_3m_4$ is the desired shadow region. The lines m_1m_4 and m_2m_3 are referred to as the *shadow lines* of m_1m_2 due to l_1l_2.

Case 2: Consider the case of a line segment l_1l_2 and a convex polygon P in a plane. We need to find the shadow region of P due to the line l_1l_2 when P lies on the visible side of l_1l_2. This region is the intersection of the shadow regions of each edge v_iv_{i+1} of P due to l_1l_2. An easier way to compute this shadow region is to find the two vertices of P, x, and y, such that the shadow region of the line segment xy due to l_1l_2 is the desired shadow region of P due to l_1l_2. The two vertices x and y are referred to as the *shadow vertices* of P due to l_1l_2. Consider the two sub cases:

- *Subcase 1:* As shown in Fig. 3(a), the entire polygon lies to the left of the line l_1l_2. That is, the extension of the line does not cut any of the edges of the polygon.
- *Subcase 2:* As shown in Fig. 3(b), only part of the polygon lies to the left of the line. That is, the extension of the line cuts some of the polygon's edges.

The determination of the shadow vertices is different in each Subcase. `Project` detects these Subcases and uses a different method in each. For Subcase 1, shown in Fig. 3(a), the shadow vertices are found as follows: the vertex of P corresponding to the maximum of the angle $\angle l_2l_1v_i$, $i = 1, 2, \cdots n$ gives the shadow vertex x, and, similarly, the vertex of P corresponding to the maximum of $\angle l_1l_2v_i$, $i = 1, 2, \cdots n$, gives the shadow vertex y.

It is slightly more involved to find x and y in Subcase 2, shown in Fig. 3(b). First, the convex hull of the points $v_1, v_2, v_3, \cdots v_n$ and l_2 is formed. Then the two vertices of the convex hull adjacent to l_2, v_1, and v_3 are considered. Of these, the vertex to the left of line l_1l_2 is considered to be the shadow vertex y and the other vertex is considered to be x. Hence, $x = v_1$ and $y = v_4$. To find the shadow region, line l_1x is extended and its intersection with the plane is found as a; similarly the intersection for the line l_2y is found as b. Thus, the shadow lines are xa and yb and the shadow region is $xaby$. Once the shadow region is formed, the edge l_1l_2, numbered as 5 in Fig. 3(b), is marked as invisible and, hence, is removed from the VL's of all the EVR's that lie inside this shadow region. Note that in Subcase 2, all of the shadow region does not lie in the visible side of the line l_1l_2. As a result, we have adjacent EVR's with identical VL's. However, the `Merge` process, to be discussed next, accounts for this situation.

`Merge`: Given an EVR list and the associated VL's, this `Merge` process searches the list for adjacent EVR's with identical VL's. Since the EVR's are actually convex polygons, two EVR's are considered adjacent if they have one edge in common. These EVR's are then merged into a single EVR, and the EVR list is updated if they have identical VL's. This step runs iteratively, each time merging the adjacent EVR's and forming new ones until no two adjacent EVR's have identical VL's.

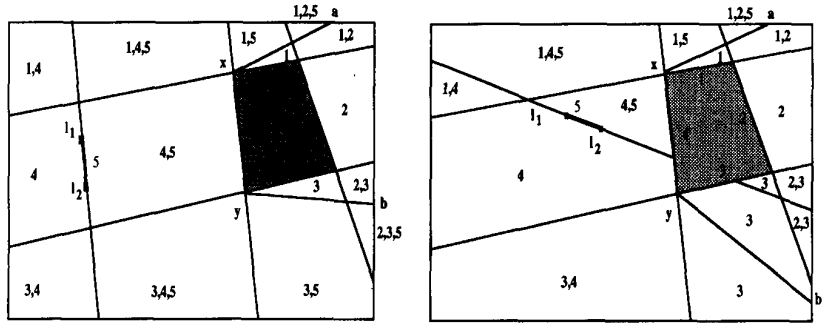


Fig. 3. (a) The shadow regions for Subcase 1. (b) The shadow regions for Subcase 2.

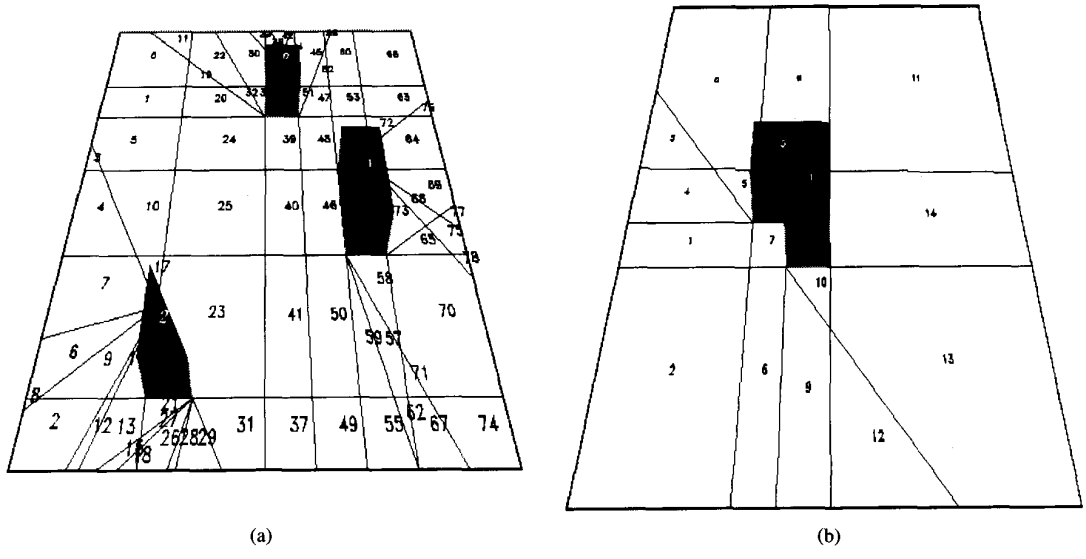


Fig. 4. (a) The complete EVR representation. (b) The EVR's of a building with a nonconvex rooftop.

A. The Algorithm Partition

Given a list of convex polygons and a viewing plane, the algorithm's divides the viewing plane into EVR's and forms their associated VL's. The algorithm initially assumes the entire viewing plane to be one EVR and then iteratively splits this plane into subregions, considering each of the polygons in turn using the process `Split`. Whenever a new polygon is considered, not only are the existing EVR's split to account for the edges' visibility, but the occlusions of the polygon's edges, due to all the other existing polygons in the plane, are also considered, using the `Project` process. The `Merge` step is run after each `Project` step to merge adjacent EVR's having the same VL's. As described before, the routine `Split`(P,LIST) takes a polygon P and an EVR list LIST, and modifies the EVR's in LIST and their associated VL's, depending on the visibility of the polygon P's edges. The `Project`(e,P,List) routine takes in an edge e, a polygon P, and an EVR list LIST and modifies the LIST and the associated VL's accounting for the occlusion of e due to the edges of P. The `Merge`(LIST) routine takes an EVR list, LIST, merges the adjacent EVR's having the same VL's, and returns the modified LIST.

The list ALL_POLYGONS contains all the given model polygons. The algorithm iteratively calls `Split` on each of these polygons to divide the viewing plane into EVR's. Also, after calling `Split` for each of the edges of the CURRENT-POLYGON, we call `Project` to handle the occlusions of these edge due to the edges of the already existing polygons in the plane. We then call `Project` to handle the occlusions of the other edges due to these edges also. After each call to `Project`, we call `Merge` to merge the adjacent regions with identical EVR's. Finally the algorithm returns with the list of EVR's describing the environment in the EVR_LIST, along with the EVRs' VL's.

Fig. 4(a) shows the resulting EVR's generated by the above process for a simulated urban environment. The environment considered here consists of three buildings. There are 79 EVR's for this given scenario.

An interesting and important question related to using this method is how many EVR's are generated using the `Partition` algorithm. If the number is very large, it is impractical to use the method. One might think that the number of distinct EVR's grows exponentially with m and n, the number of polygons and the number of sides of each polygon,

Algorithm `Partition`:

Input: A set of convex polygons in a plane.
Output: A set of EVR's and their associated VL's.

```
1. Initialize
   1.1 Initialize the EVR_LIST to
       contain one region, the viewing plane.
   1.2 Set the VL of this region to NULL.
   1.3 Initialize the USED_POLYGONS
       list to NULL.
2. For ( i = 1 to i < No_of_ALL_POLYGONS )
   do
   2.1 Set CURRENT_POLYGON =
       ALL_POLYGONS(i)
   2.2 Call Split(CURRENT_POLYGON, EVR_LIST)
   2.3 For ( j = 1 to j < No_of_USED_POLYGONS)
       do
       2.3.1 For ( k = 1 to k < No_of_edges_of
          (CURRENT_POLYGON) )
          do
          2.3.1.1 Call Project
                [EDGE_of(CURRENT_POLYGON,k),
                USED_POLYGONS(j),EVR_LIST)
          2.3.1.2 Call Merge(EVR_LIST)
          endo
       2.3.2 For ( k = 1 to k < No_of_edges_of
                (USED_POLYGONS(j)) )
          do
          2.3.2.1 Call Project
                (EDGE_of(USED_POLYGONS(j),k),
                CURRENT_POLYGON,EVR_LIST)
          2.3.2.2 Call Merge(EVR_LIST)
          endo
       endo
   2.4 USED_POLYGONS (No_of_USED_POLYGONS)
       =CURRENT_POLYGON
   2.5 Increment No_of_USED_POLYGONS
   endo
3. Exit
```

respectively. In Appendix B, we derive an upper bound on the maximum number of EVR's generated and show that this is polynomial in m and n, $O(n^2m^4)$. However, this is a very loose upper bound and, in practice we can never generate this many regions, as Appendix B explains.

B. Estimating the Range of Orientations

For each model edge in the VL of an EVR, the range of robot orientations for which this model edge is visible in the EVR is also computed and stored. Below is an efficient method to compute the range. Given an EVR and a model edge e_i in the VL of the EVR, we need to find the lower bound, $\theta_{\min}$, and the upper bound, $\theta_{\max}$, of the robot's orientation angles for which this edge e_i is visible in the EVR. Let the vertices of the EVR be v_i, $i = 1, \cdots n$. Let p_1, p_2 be the end points of e_i. Then it is easy to show that $\theta_{\min}$ is the minimum of the angles made by the lines joining p_2 to $v_i (i = 1, \cdots n)$ and that $\theta_{\max}$ is the maximum of the lines joining p_1 to v_i, $i = 1, \cdots n$. Hence, given an EVR and a model edge e_i in its VL, the maximum and minimum orientation angles are estimated by considering only the EVR vertices.

C. Generalizations

In the case of buildings with nonconvex rooftops, the nonconvex polygons representing the rooftops' projections onto the xy-plane are decomposed into a set of adjacent, component convex polygons. The convex decomposition of nonconvex polygons is a well studied problem [23], [24]. Decomposing a simple polygon into nonoverlapping components can be done with or without introducing additional vertices, which are commonly called *Steiner points*. Since we are to process these polygons further, we are only interested in decompositions that do not introduce additional points. Green [24] developed an $O(n \log n)$ algorithm that finds a decomposition within four times of the minimum decomposition when Steiner points are disallowed. Our work uses this algorithm to decompose the nonconvex polygons composing up the rooftops into convex subparts. The extra edges added in the process of converting a nonconvex polygon to adjacent convex polygons are considered *dummy* edges. The `Partition` algorithm is then modified so that the dummy edges are not used in the `Split` process and their occlusions are not considered in the `Project` process. Essentially they do not contribute to the VL's of the EVR's but do serve to make the algorithms developed in this work even in the nonconvex cases. Fig. 4(b) shows the EVR's of a building with a nonconvex rooftop. Here, the self occlusions of the edges of a nonconvex polygon are handled by decomposing the polygon into component convex polygons, and dealing with their mutual occlusions using the `Project` process.

When the buildings are of unequal heights, the `Project` process is more complicated. In forming the shadow region of an edge e onto a polygon P, it is insufficient to consider just two polygons' vertices in forming the shadow region, since the heights of the edge and the polygon are different. So, the `Project` process is modified as follows: the two shadow vertices x and y of P are computed as before, and the two end points of the edge e, e_1, and e_2, are then projected onto all the edges of P lying between x and y and on the polygon's *invisible* side, that is, the side of P where e is not visible. Since e and P are of different heights, the projections from e_1 intersect the viewing plane, forming a convex polygon P_1, and similarly, the projections from e_2 form a convex polygon P_2. The intersection of these two convex polygons, P_1 and P_2, is itself a convex polygon P_3, which gives the required shadow region where e is invisible due to occlusions from P. Fig. 5 illustrates these ideas. Forming the intersection of two convex polygons is a well studied problem. Preparata [21] reviews the existing methods. Once the shadow region is computed, the edge e is then marked as being invisible in all the EVR's lying in this region. The EVR's formed by this technique are always convex regions when all the buildings are of equal height. However, when the buildings are of different heights, it is possible that due to the `Project` process, the intersections with the viewing plane cause some of the EVR's

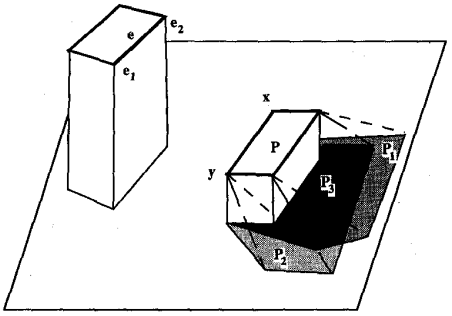


Fig. 5. The shadow region of an edge and a polygon of different heights.

to become nonconvex. The nonconvex EVR's are detected during the `Merge` process. The `Merge` process is modified so that it does not merge two adjacent EVR's with identical VL's if the resulting EVR will be nonconvex. Instead the process maintains the information that these two EVR's are parts of the same EVR. In essence, we represent the nonconvex EVR's by their component convex parts. This convexity property of the EVR's is useful in making the computations in the EVR formation process easier.

When the entire rooftop of a building is not one *flat* (parallel to the ground plane) polygon, it is decomposed into multiple flat, convex planar polygons of different heights and each of these is considered a separate polygon before applying the `Partition` algorithm. If the rooftop of the building is not *flat*, i.e., it is sloping, then the `Project` process needs to be modified. It is no longer sufficient just to consider two vertices for each edge in forming the shadow region. Instead, the projection of the entire sloping plane of the rooftop onto each of the edges of the other rooftops needs to be considered.

The world model that we considered consisted of building with *flat* rooftops that were of the same height. However, some of the buildings were nonconvex and had to be decomposed to component convex subparts. The extensions of the algorithm to unequal and nonflat rooftops has not been currently implemented.

III. Position Estimation

Given a world model, we form the EVR description of the free space as discussed above. This section describes a method to estimate the robot's position and pose using such a description. Talluri and Aggarwal present the initial results of testing this method [25], [26] using computer simulations.

We use a two-stage constrained-search strategy to isolate the robot's position and pose. First, we select a small set of EVR's likely to contain the robot location by using a modified Hough transform strategy for transform clustering. To isolate the EVR containing the robot's location, we form an interpretation tree of all possible model-image feature correspondences for each EVR. These trees are then searched using the geometric constraints imposed by the EVR's and the image features. Each tree node is an association between a model and an image feature. A path from the root to the leaves of the tree gives a set of possible associations between each of the image features and a model feature. If no such path exists, then this EVR is discarded as not containing the robot location and the next EVR in the candidate set is considered. Finally, the correct EVR containing the robot location is isolated, and a set of model-image feature correspondences is established. All these correspondences are then used in a least squares paradigm to compute the transformation matrix $\mathcal{T}(x, y, \theta)$ and, thus, to estimate the robot's position (x, y) and pose θ.

A. Transform Clustering for Model-Image Feature Correspondence

Previous research used the generalized Hough transform and its related parameter hashing techniques to perform transform clustering to isolate the transformation that maps the model features to the image features [8], [10], [27], [28]. The problems associated with using the Hough transform approach to transform clustering are that large transform clusters may occur randomly. If these clusters are as large or larger than those due to the correct transform, the estimation procedure that relies only on the Hough transform will be erroneous. If the number of buckets is increased, then the possibility of random large clusters is alleviated but the number of computations grows rapidly. Grimson [6] summarizes these problems with the generalized Hough transform.

This research suggests a method to reduce the problems associated with the Hough transform approach to transform clustering [29] by using a partition of the parameter space, which is not necessarily uniform. The partition is, in fact, *intelligent* and uses *a priori* model information. Recall that the EVR's partition the robot's free space into distinct, nonoverlapping regions. In its VL, each EVR has an associated list of model features visible in the region. Also, the range of θ's for which each feature is visible is stored in the EVR. Hence, the EVR's effectively partition the entire parameter space of (x, y, θ), taking into account the visibility of the model features. Thus, given a set of image features to be used in the position estimation, the idea is: 1) to hypothesize a correspondence between all pairs of model and image features; 2) to compute the range of possible transformations for this hypothesis; and 3) to vote in all the EVR's where this transformation is feasible. Selecting the EVR's with the largest number of votes produces a reduced set of EVR's that are most likely to contain the robot's location.

Note that each of the image features is a 2-D line in the image plane and that each of the model features is a 3-D line segment in the WCS. The transformation that takes the model feature into the image feature under perspective projection contains three degrees of freedom, as discussed before, and is the required robot's position and pose. Thus, we need at least two model-image feature correspondences to be able to solve for the three unknowns since each correspondence gives two equations using perspective projection. However, by using one 2-D to 3-D line correspondence, we can solve for the rotation θ and arrive at a constraint on the translation (x, y) which is in the form of a straight line $\mathbf{L}$: $ax + by + c = 0$ in the xy-plane. For this procedure we use a method similar to one described by Liu *et al.* [30]. This procedure is detailed in Appendix C.

Thus, for each hypothesized model-image feature pairing, we search the EVR list and vote in an EVR if the following

conditions are satisfied: 1) the model feature is present in the EVR's VL; 2) the θ computed from the transformation lies within the range of angles for which the model feature is visible in the EVR; and 3) the straight line **L** representing the constraint on the translation intersects the EVR. After considering all the pairings, we search for clusters in the EVR's by considering the EVR's with a large number of votes as the list of EVR's to be considered as candidates for position estimation. The advantage of this approach is that the noise immunity provided by the Hough transform approach gives a robust method to help prune the list of EVR's and to isolate a smaller subset most likely to contain the robot's position. From our experiments, we find this transform clustering method to be quite effective.

B. Interpretation Tree Search

By using the transform clustering technique described above, we now have a small set of candidate EVR's likely to contain the robot's location. Instead of using a traditional backtrack, depth-first search technique to search the interpretation trees corresponding to the candidate EVR, we exploit the tree's repetitive nature and the EVR paradigm to evolve a more efficient search strategy. We separate the geometric constraints into two types: unary constraints and coupled constraints. Unary constraints are those that need to be satisfied by each of the model-image feature correspondences. Coupled constraints are those that need to be satisfied by a set of model-image feature correspondences.

The following unary constraints need to be satisfied by a model-image feature correspondence:

- **Rotation Constraint:** The computed rotation angle θ should lie within the range of possible orientation angles stored for that model feature in the EVR's visibility list.
- **Translation Constraint:** The straight line **L** representing the constraint on the translation vector should intersect the EVR.

The following coupled constraints need to be satisfied by a set of model-image feature correspondences:

- **Rotation Consistency Constraint:** All the rotation angles computed by the correspondences in the set should be consistent. This is tested by checking if the rotation's standard deviation is less than a certain threshold value.
- **Ordering Constraint:** The left-to-right order in which the set of image features are visible in the image plane should be maintained by the corresponding world model features. Once the robot's rotation is computed, the ordering of the world model features can be verified from the stored world model.
- **Transformation Consistency Constraint:** The final position estimated by combining all the correspondences using a least squares technique should lie within the EVR, and the estimated pose should be valid in the EVR.

For each of the candidate trees, we initially propagate the unary constraints and prune large subtrees from the search space. The sparse tree so formed is then searched using the coupled constraints. The coupled constraints are then applied to determine if any of the paths results in a valid set of model-image feature correspondences.

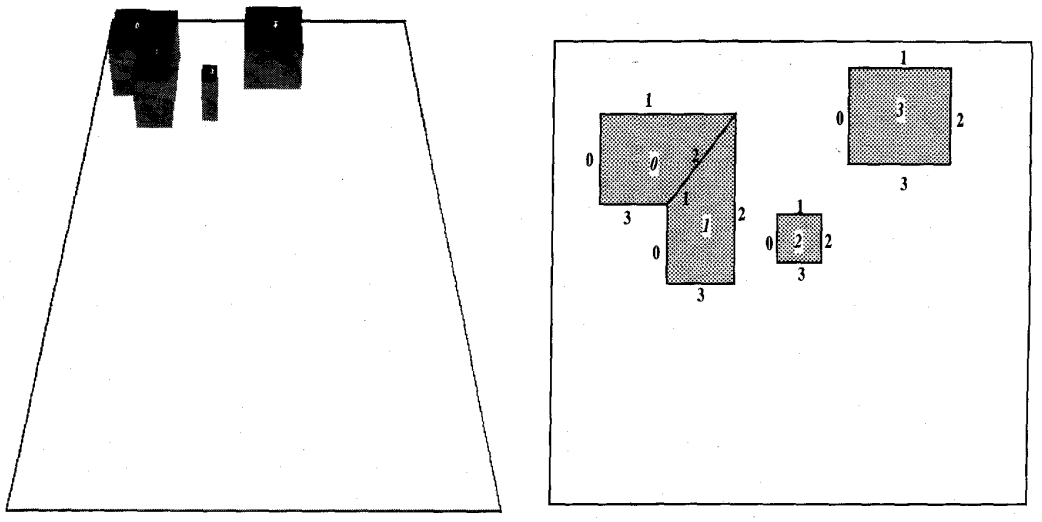
A path from the tree's root to its leaves gives a set of possible associations between each of the image features and a model feature. If no such path exists, then this EVR is discarded as not containing the robot's location and the next EVR in the candidate set is considered. Finally, the correct EVR containing the robot's location is isolated and a set of model-image feature correspondences is isolated. Note that we need only two model-image feature correspondences to compute the three unknowns in the robot's position and pose. For better noise immunity when there are more than two correspondences in the path, we use a least squares paradigm to combine the correspondences and compute the transformation matrix $\mathcal{T}(x, y, \theta)$ and, hence, estimate the robot's position (x, y) and pose θ. We find this technique of searching the interpretation tree to be much more efficient, and computationally less expensive [31] than a traditional backtrack search technique suggested by previous researchers [5], [7], [32].

Note that our research considered the *drop-off* problem; that is, the robot does not have an *a priori* estimate of its current location. However, if the robot can operate in a *boot-strap* mode or use its odometry, and thus maintain a rough estimate of its position and pose, this estimate can reduce the search space significantly. The current estimate can be utilized to isolate the robot's position to within a few EVR's so that only these EVR's need to be considered when searching for the exact location Also, if the robot can identify landmarks (some of the 3-D features), this ability can be used to isolate the robot's position to within only those EVR's containing these landmark edges in their VL's.

C. Noise Effects

The noise sources that can affect the position estimation are the errors in image feature extraction and those in the world model description. If the environment of the robot contains obstacles, such as trees and telephone poles, that do not belong to the model and occlude some of the model features, it is possible that the feature detector used to extract the rooftop lines may: 1) miss some of the model lines; 2) extract false lines; and 3) incorrectly estimate the line parameters of the extracted lines. Also, if the world model description is incomplete (for example, if some of the buildings parts are missing or if they are stored incorrectly in the world model), then the above phenomenon may also occur. We wish to make our technique robust to such occurrences.

Since there are only three unknowns in the robot's position and pose, (x, y, θ), we only need two correct model-image feature correspondences to solve for these unknowns. Thus, even if some of the image features corresponding to the model features are missing, we can still estimate the robot's position and pose if we find at least two correct model-image feature correspondences. Thus, some lines being missed by the feature extractor does not pose a serious problem. To account for the noise in the edge detector output in the feature extraction stage, we also use a least square fit to the edge



(a) (b)

Fig. 6. (a) The robot's environment. (b) The labels of the world model edges (top view).

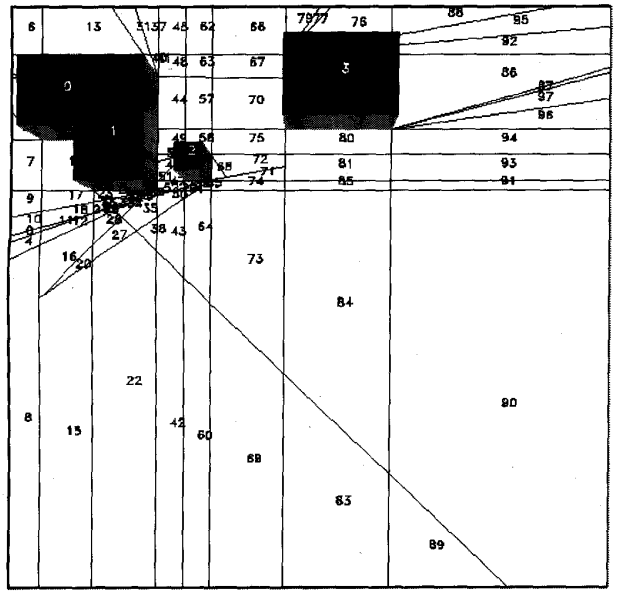


Fig. 7. The EVR representation of the environment.

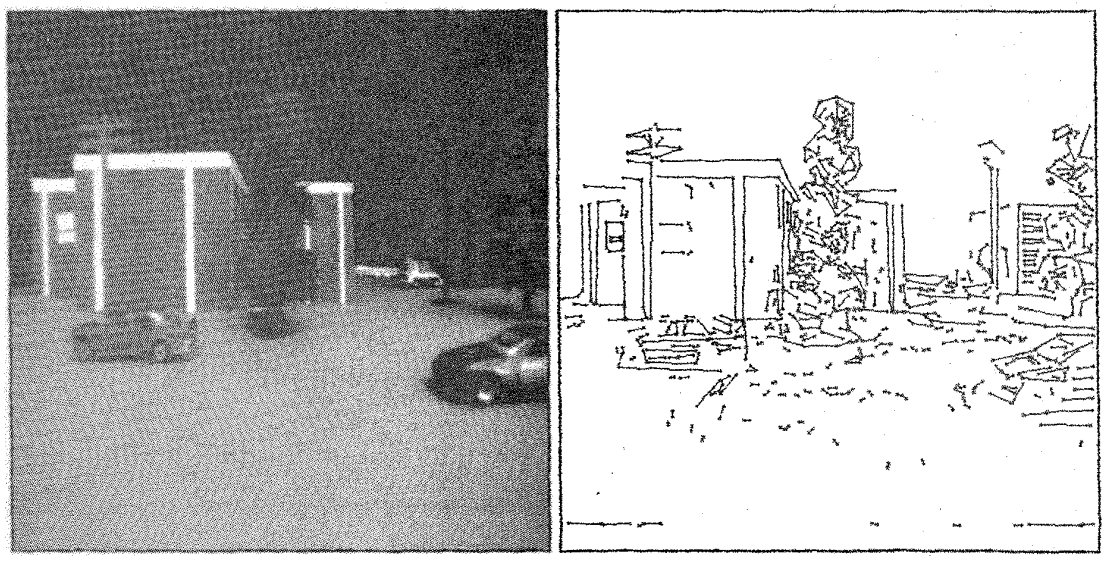
TABLE I
TWO OF THE EVR'S AND THE CORRESPONDING VL'S

EVR Number	*No. of Edges in the VL*	*Visibility List*			
		obj.no	*obj.side*	θ_{min} deg	θ_{max} deg
42	7	0	3	-75.437	22.133
		1	2	-47.011	30.000
		1	3	-75.437	30.000
		2	0	-30.000	50.599
		2	3	-30.000	66.330
		3	0	-19.771	75.219
		3	3	-18.142	91.655
90	6	1	3	-120.000	-8.565
		1	2	-120.000	0.398
		2	2	-118.506	1.410
		2	3	-118.598	-0.639
		3	2	-103.888	30.000
		3	3	-109.015	30.000

detector output to form the line parameters. This technique has the effect of smoothing out the noise effects. We also *dilate* the EVR by making its boundaries *fuzzy*; that is, the EVR extent in the xy plane is made more flexible. Similarly, the range of possible orientations for each edge in the VL of the EVR is also made more flexible. Thus, the basic idea is to introduce tolerance bands to handle the effects of noisy sensor data.

To deal with spurious image features, we propose a technique similar to the one suggested by Grimson [8]. The idea is to introduce *null features* below each node in the search tree. Pairing a sensor feature with this null feature is equivalent to discarding the sensor feature as inconsistent with the model. The null feature thus acts as a *wild card* in the match. Therefore, pairing any sensor feature with the null feature is considered to be consistent; however, this pairing does not contribute to the transform computation. By using this null branch approach, we associate all the true image features to the model features and the spurious image features to the null branches.

We find that noise effects are more significant on short sensor features of less than ten pixels. To deal with this, if the line is too short, we do not consider it in the hypothesis stage. Since our system can handle missing features, removing these short lines does affect the position estimation strategy. The tolerance bands introduced in the EVR's are also made functions of the line lengths so that the shorter lines have larger tolerance bands. We also weigh the contribution of the image features by length, considering the longer features more reliable. In combining multiple matches in the final stage, which computes the actual robot's position and pose, we use a weighted least squares estimate.



(a) (b)

Fig. 8. (a) The image used by the robot in position estimation. (b) The output of the line detector.

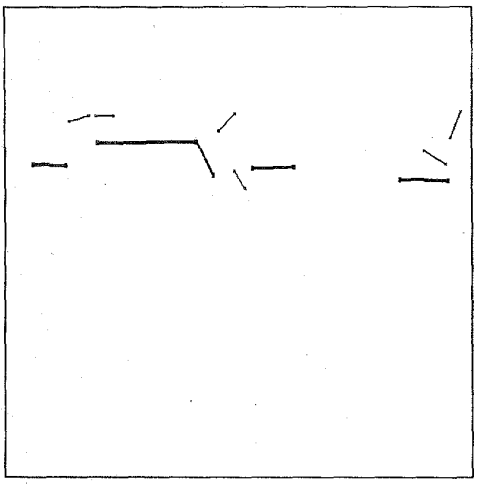


Fig. 9. The lines used as image features. The lines in bold are isolated as corresponding to the model features and the plain lines as noise features.

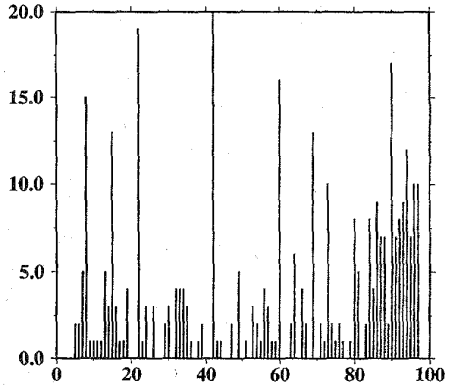


Fig. 10. EVR number versus the number of votes.

These techniques are implemented and tested using a scale model of a real airport. The results are quite promising, and the robot's position and pose are estimated to a high degree of accuracy, even in the presence of noise. Section IV discusses the results.

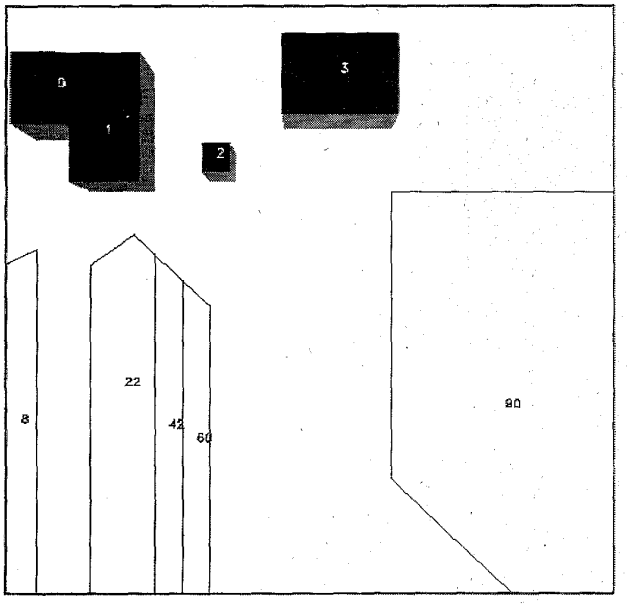
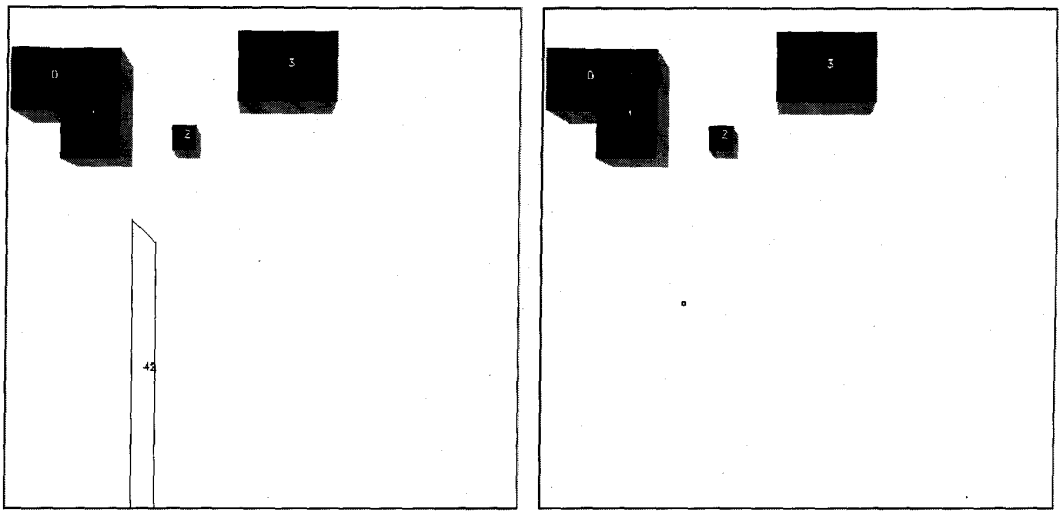


Fig. 11. The candidate set of EVR's isolated by the Hough Transform.

IV. Experimental Results

The technique of estimating the robot's position and pose robot using the EVR's has been implemented and tested using a scale model of a real airport environment. The scale model was built using the blueprints of the Austin Executive Park airport. Fig. 6(a) shows the environment. The airport contains three polyhedral buildings, one having a nonconvex rooftop. Here the 3-D descriptions of the rooftops are assumed to be known to the robot as the world model. Fig. 6(b) shows the



(a) (b)

Fig. 12. (a) The EVR isolated by the search as the one containing the robot's position. (b) The estimated robot location.

labeled polygons constituting the rooftops and their edges. Fig. 7 shows the EVR representation of this description. In forming this description, the nonconvex rooftop is partitioned into two component convex polygons, such that the scene now contains four rooftop polygons. Table I lists two of the EVR's and their VL's. For the given scene there are 98 EVR's.

We used a CCD camera to capture an image of the scale model. Fig. 8(a) shows one such image used in the position estimation process. The camera's internal calibration parameters, the lens center, (u_0, v_0) and the scale factors (S_y, S_z) are first calculated. Using the rooftop lines, we need to establish a correspondence between this image and the model shown in Fig. 6(a) and estimate the robot's position and pose.

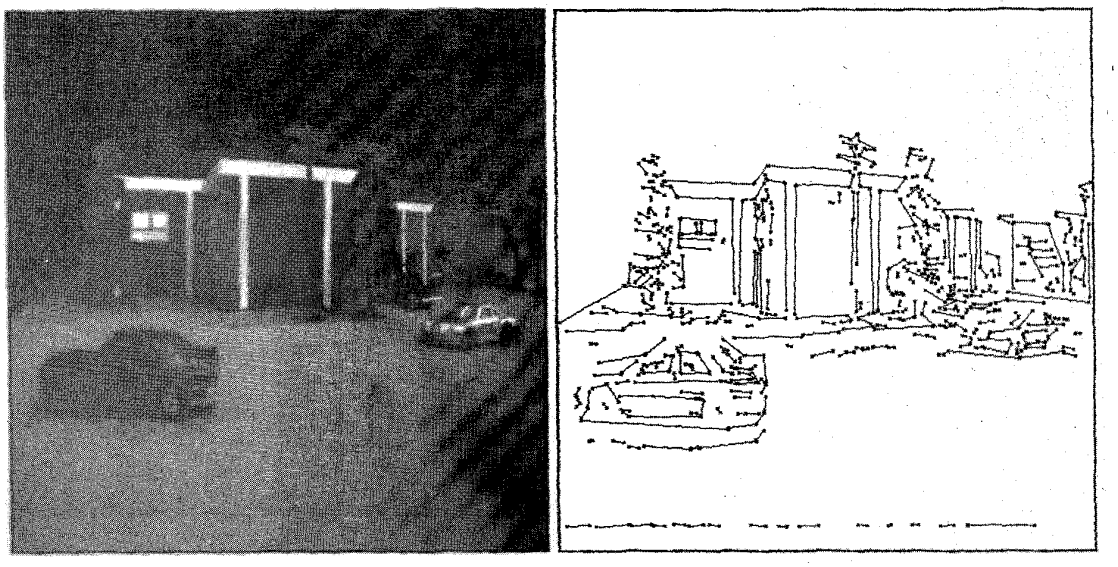
The image contains a significant amount of noise arising from obstacles that are not part of the model, such as trees and telephone poles. A Canny edge detector detects the edges from this image. We chain adjacent pixels and fit lines to them and then threshold all lines less than 20 pixels long. Fig. 8(b) shows the output from this operation. We use a simple *rooftop lines* extraction process by scanning the lines in Fig. 8(b) from the top and selecting only the *top-most* lines. We then remove the lines that lie completely within the shadow of a selected line. This process yields a set of image features used in the localization process as shown in Fig. 9. Note that due to the obstacles and errors in feature detection, some of the rooftop lines are missing and a number of spurious lines are included as image features. However, we find that by using the position estimation scheme described above, we can isolate the spurious features from the true image features and accurately estimate the robot's position and pose. Eleven image features in this scene are used in the position estimation. In Fig. 9, the true image features are indicated by bold lines and noise features by plain lines as isolated by the constrained search technique.

We use the transform clustering technique outlined before to initially select the EVR's likely to contain the robot's position. We hypothesize a correspondence between each of the eleven lines extracted from the image in Fig. 9 and each of the model edges in Fig. 6(a). For each hypothesis we compute the range of possible transformations and vote in all the EVR's satisfying this transformation. Fig. 10 plots the EVR number versus the number of votes for each EVR. We select the five EVR's with the largest number of votes as the likely candidates to contain the robot's position. Fig. 11 shows these five EVR's. The actual robot location is in EVR number 42. This EVR has the highest number of votes, 20, and hence is included in the list of likely candidates.

For each of the five candidate EVR's, an interpretation tree is formed of the possible pairings between the image edges and the world model edges in the EVR's VL. We also use the null branch at each tree node to handle noise features. These trees are then searched by first propagating the unary constraints and then using the coupled constraints. We find that the tree associated with EVR 42 is the only one with a path from the root to the leaf. Fig. 12(a) shows this EVR. By combining all the model-feature correspondences dictated by this path using a least squares technique we estimate the robot's position and pose to be (874.933, 411.292) and its pose to be 1.026 233°. Fig. 12(b) shows the robot's location. These estimates are found to be quite close to the true values. Thus, the path correctly associates the true image features with the model features and the noise features with the null branches.

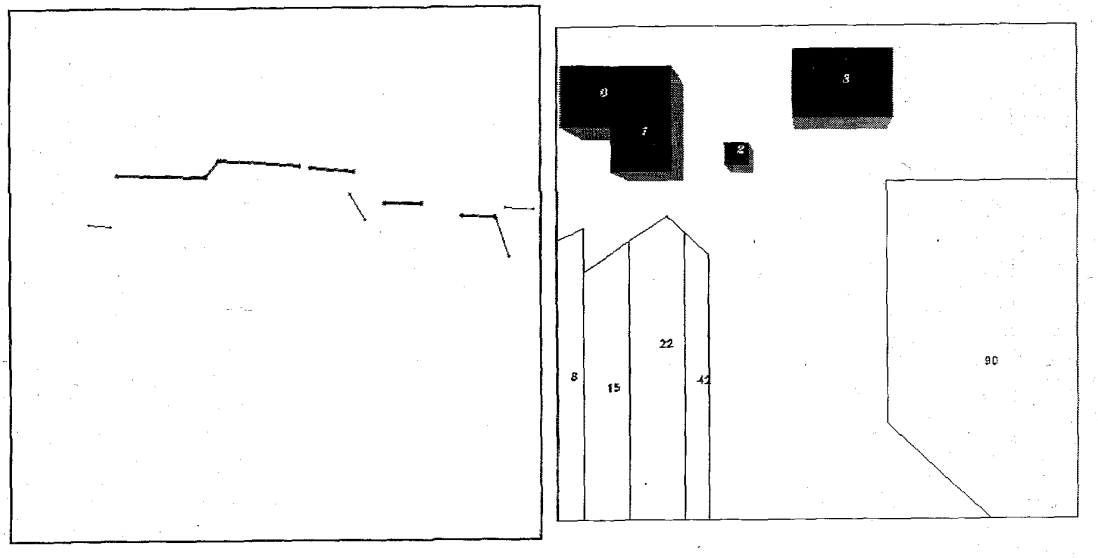
A number of test runs are performed using the given world model description and different robot positions. Table II shows the results. In most cases the estimation procedure succeeds in isolating the robot's position and pose quite accurately.

Figs. 13–15 show the results of the position estimation process for another test run. A point worth noting in this test run is that, some of the image features are broken due to the partial occlusions by the telephone pole. However, since the constrained search algorithm allows for multiple image features to match to the same model feature, both these image features are labeled as corresponding to the same model feature and hence resulting in an accurate position and pose estimate.



(a) (b)

Fig. 13. (a) The image used by the robot in position estimation. (b) The output of the line detector.



(a) (b)

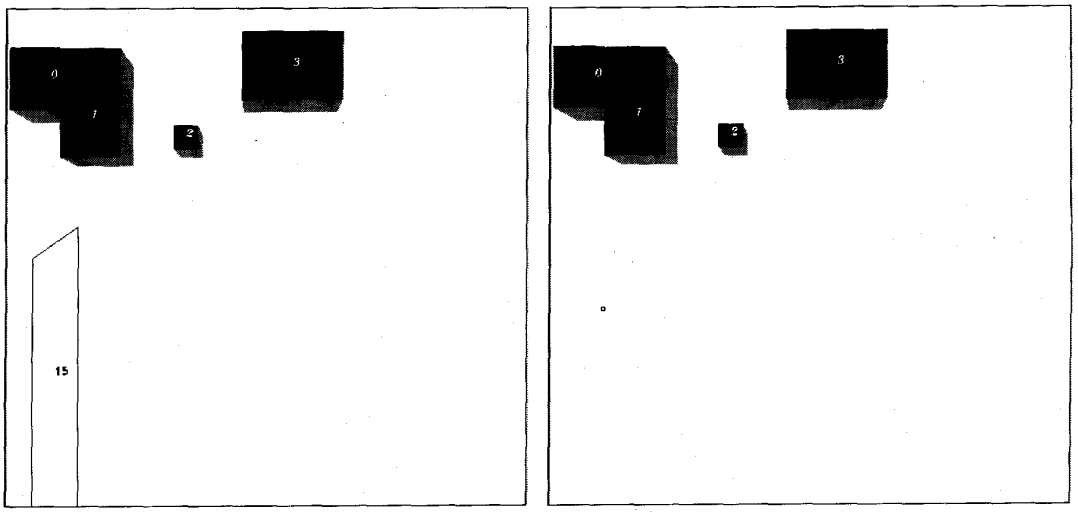
Fig. 14. (a) The lines used as image features. The lines in bold are isolated as corresponding to the model features and the plain lines as noise features. (b) The candidate set of EVR's isolated by the Hough Transform.

Occasionally the search strategy isolates more than one EVR as containing the robot's location. In such cases, we translate the robot by a known extent to a different position and consider another image for the position estimation. The search results at this second stage must now satisfy the additional constraints of the first position. These additional constraints help eliminate the ambiguity in the robot's position.

V. Conclusion

This paper considers the problem of establishing correspondence between a stored 3-D model and a 2-D image of it. We present an approach to solving this problem in the context of autonomous navigation of a mobile robot in an outdoor urban, man-made environment consisting of polyhedral buildings. The 3-D descriptions of the lines constituting the buildings' rooftops are assumed to be given as the world model. The robot's position and pose are estimated by establishing a correspondence between the model features and the straight line features extracted from the images acquired by the robot.

The model-image feature correspondence problem is formulated as a two-stage constrained-search problem. Geometric visibility constraints are used to reduce the search space of possible model-image feature correspondences. Techniques are presented for effectively deriving and capturing these visibility



(a) (b)

Fig. 15 The EVR isolated by the search as the one containing the robot's position. (b) The estimated robot location.

constraints from the given world model by developing a novel, intermediate representation of the environment, called the *Edge Visibility Regions* (EVR's). Using a modified Hough transform technique, a small set of candidate EVR's most likely to contain the robot's location are isolated. For each of these candidate EVR's, an interpretation tree is formed hypothesizing all possible pairings between the world model features and the image features. This tree is then searched for a consistent set of matches between the world model features and their images. The tree is pruned using the geometric constraints between these features captured by the EVR representation of the environment. The position estimation technique presented is shown to be robust and accurate even in the presence of errors in feature detection, incomplete model description, and occlusions. Experimental results of testing this approach using a model of an airport scene are presented.

Future work includes extending the approaches developed in this research to buildings that are not necessarily polyhedral and also exploring the applicability of these techniques to other areas of computer vision research, such as model-based object recognition.

TABLE II
THE SEARCH RESULTS

Actual Position	Actual Pose deg	Estimated Position	Estimated Pose deg	EVR No.
(875,410)	0	874.93,411.29	1.03	42
(750,1010)	-30	752.45,1016.21	-32.02	90
(900,50)	5	901.74,49.84	4.87	8
(800,800)	-5	798.63,800.12	-4.58	84
(1320,1350)	-15	1324.27,1347.46	-16.14	89
(350,1400)	-90	352.23,1404.67	-92.27	94
(460,1400)	-75	460.15,1406.14	-78.04	91
(425,1450)	-85	428.57,1454.21	-88.13	93
(1350,900)	-15	1351.12,896.14	-14.21	83
(1475,600)	0	1480.11,605.26	1.02	69

APPENDIX

A. Estimating the Number of EVR's

In this section, we derive a loose upper bound on the maximum number of EVR's generated by the *Partition* algorithm given mn sided convex polygons. We show that this upper bound is polynomial in m and n, $O(n^2m^4)$. The derivation is based on induction, and is similar to that outlined by Ikeuchi and Kanade [17] in the context of aspect graphs for object recognition.

Consider the case of kn-sided convex polygons in a plane. Let the total number of lines in the plane be given by $L(k)$. We have

$$L(k) = L(k-1) + n + 2 \cdot n \cdot (k-1) + 2 \cdot n \cdot (k-1) \quad (1)$$

where the first term indicates the number of lines already existing after considering $(k-1)$ polygons; the second term indicates the number of lines drawn by extending the kth polygon's edges; the third term accounts for the shadow lines drawn from the n edges of the kth polygon onto the other $k-1$ already existing polygons; and, finally, the fourth term accounts for the shadow lines drawn from each of the n edges of the existing $k-1$ polygons onto the kth polygon. Recall that each polygon edge contributes two *shadow lines* for each existing polygon lying on the visible side of the edge. Solving (1) we have

$$\begin{aligned} L(k) &= kn + 4n[(k-1) + (k-2) + \cdots 1] \\ &= kn(2k-1). \end{aligned} \quad (2)$$

Let $T(m)$ denote the maximum number of EVR's generated when we have m n-sided convex polygons in a plane under perspective projection. Note that $T(0) = 1$. Consider the situation where are $m-1$ polygons already exist in the plane. The maximum number of lines in the plane is $L(m-1) = n(m-1)(2m-3)$. Now when we consider the mth polygon, we need to perform three steps that increase the number of lines and thus the EVR's. First, each edge of the mth polygon

is extended, which gives n lines. Second, for each of the n edges of this mth polygon, we draw two shadow lines onto each of the other $m-1$ polygons already in the plane; this gives $2n(m-1)$ lines. Finally, for each of the n edges of the $m-1$ polygons, we draw two shadow lines onto the mth polygon, which also gives $2n(m-1)$ lines. So, we add a total of $n+4n(m-1)$ new lines by adding the mth polygon.

If we assume that each of these $n(4m-3)$ lines is added successively, the number of EVR's grows after adding each new line. We can derive the maximum number of EVR's by induction. Let $T(m)_k$ denote the number of EVR's after drawing the kth line. Since we start with $T(m-1)$ EVR's and $L(m-1)$ lines, after drawing the first additional line, the number of EVR's, $T(m)_1$, is

$$T(m)_1 = T(m-1) + L(m-1) + 1$$

since this line is cut by the existing $L(m-1)$ lines into $L(m-1)+1$ segments (maximal case). These segments will divide each of the $L(m-1)+1$ regions into two regions, thus adding $L(m-1)+1$ new EVR's. Proceeding in this fashion,

$$\begin{aligned} T(m)_2 &= T(m)_1 + L(m-1) + 2, \\ &\quad T(m)_{n(4m-3)} \\ &= T(m) \\ &= T(m)_{n(4m-3)-1} + L(m-1) + n(4m-3) \end{aligned}$$

and

$$\begin{aligned} T(m) &= T(m-1) + n(4m-3)L(m-1) + \sum_{i=1}^{n(4m-3)} i \\ &= T(m-1) + \frac{[n(4m-3)]^2}{2} + \frac{n(4m-3)}{2} \\ &\quad + n(4m-3)[n(m-1)(2m-3)] \\ &= T(m-1) + O(n^2m^3) + O(n^2m^2) + O(mn) \\ &= O(n^2m^4). \end{aligned}$$

It is worth pointing out that although the maximum number of regions suggested by this derivation is $O(n^2m^4)$, in practice it is very unlikely that so many EVR's will be generated. First, the derivation assumes that whenever a new line is drawn, it cuts all the existing lines in the plane, which very rarely happens. Second, the shadow lines are considered drawn onto all the existing polygons at any given time. In practice, however, we only need to draw the shadow lines onto the polygons lying on the edge's visible side. Since we consider convex polygons, the situation where all the polygons lie on all the edge's visible sides is an impossible one. Also, the *Merge* process is quite effective and significantly reduces the number of regions, particularly as we consider an increasing number of polygons. After accounting for all these factors, we find the number of regions to be much smaller than $O(n^2m^4)$. However, this upper bound serves the useful purpose of showing that the number of EVR's, in the worst case, is still polynomial in n and m.

B. Position Estimating Using a 3-D to 2-D Line Correspondence

This section derives a method to estimate the pose and to arrive at a constraint on the robot's position using a 3-D to 2-D line correspondence. The method is similar to the one described by Liu *et al.* [30].

Let $oxyz$ represent the world coordinate system (WCS) in which the 3-D descriptions are given and let $o'x'y'z'$ represent the robot coordinate system (RCS). Let a $p=(x, y, z)^T$ be a point in the WCS and let $p'=(x', y', z')^T$ be the point after being transformed to the RCS. So we have $p'=Rp+T$, where $R=R_{z(-\theta)}R_{y(\phi)}$. $R_{z(-\theta)}$ is an unknown rotation about the z-axis of the WCS and $R_{y(\phi)}$ is a known rotation about the y-axis of the WCS to account for the tilt angle of the camera. Note that the roll angle $\psi=0$. So R has only one unknown θ. T denotes the translation vector $[T=T(x, y, Z_H)]$. It has two unknowns x, and y, because the Z_H translation, which is the height of the robot above the ground plane, is measurable and thus a constant.

Let OYZ denote the coordinate system of the image plane. The origin of this 2-D coordinate system is in the image plane's lower left corner. Let S_Z and S_Y denote the scale factors which account for the pixel spacing and the focal length along the image plane's rows and the columns, respectively. Let Y_0 and Z_0 denote the point in thee image plane where the optical axis intersects the image plane. These four parameters, S_Y, S_Z, Y_0, Z_0, are the camera's internal calibration parameters. These parameters are calculated using a calibration procedure. Then a 3-D point $p'=(x', y', z')^T$ in the RCS has as its image $P=(Y, Z)$ and they are related by the perspective projection as $Y=-S_Y(y'/x')+Y_0$ and $Z=-S_Z(z'/x')+Z_0$.

If we represent the 3-D line l in the WCS in the parametric form, $l: \vec{p}=\vec{n}t+\vec{p}_0$, where $\vec{n}=(f_1, g_1, h_1)^T$ is the direction of the line and $\vec{p}_0=(x_0, y_0, z_0)^T$ is a point on the line. The image of this line in the image coordinate system, the image feature, can be expressed as $L: Z=\alpha Y+\beta$. Using the perspective projection relations, we have $L: S_Z(z'/x')+Z_0=\alpha S_Y(y'/x')+Y_0\alpha+\beta$. This can be expressed as $M: Ax'+By'+Cz'=0$, where A, B, C are known constants and satisfy the constraint $A^2+B^2+C^2=1$. This is the equation of the projection plane containing the 3-D line l expressed in the RCS. The vector $\vec{N}=(A, B, C)^T$ is the normal to the plane M. In the RCS, we can express the direction of the 3-D line as $\vec{n'}=R\vec{n}$.

Since the 3-D line is always perpendicular to its projection plane, we have $\vec{N}\cdot\vec{n'}=0$ or

$$\vec{N}\cdot R\vec{n}=0. \tag{3}$$

Now the rotation matrix R has only one unknown, θ, since ϕ is assumed to be known, we can solve (3) for θ. There are two values of θ that satisfy this, given by θ_1 and θ_2.

Once we have the rotation matrix R, we can solve for a constraint on the translation $T(x, y)$ using this R. Note that any point on the 3-D line in the RCS must lie on the projection plane of this line. So the vector from the origin (optical center) to this point must be perpendicular to the normal of the

projection plane. So, for a known point p_0 on the 3-D line in the WCS, we have the transformed point $p_0' = Rp_0 + T$ in the RCS and $\vec{N} \cdot [Rp_0 + T] = 0$, where $p_0 = (x_0, y_0, z_0)$ or

$$\vec{N} \cdot T = -\vec{N} \cdot Rp_0. \tag{4}$$

From (4) and the known values of θ we have constraints on the translation of the form $Ax + By = s_1$ corresponding to θ_1 and $Ax + By = s_2$ corresponding to θ_2.

REFERENCES

[1] I. J. Cox and G. T. Wilfong, *Autonomous Robot Vehicles.* New York: Springer-Verlag, 1990.

[2] R. Talluri and J. K. Aggarwal, "Position estimation techniques for an autonomous mobile robot—A review," in *The Handbook of Pattern Recognition and Computer Vision,* C. H. Chen, L. F. Pau, and P. Wang, Eds. Singapore: World Scientific, 1993, pp. 769–801.

[3] C. Fennema and A. R. Hanson, "Experiments in autonomous navigation," in *Proc. 10th Int. Conf. Pattern Recognition,* Atlantic City, NJ, 1990, pp. 24–31.

[4] R. Talluri and J. K. Aggarwal, "Position estimation for a mobile robot in an outdoor environment," *IEEE Trans. Robotics Automat.,* vol. 8, pp. 573–584, Oct. 1992.

[5] R. M. Haralick *et al.,* "Pose estimation from corresponding point data," *IEEE Trans. Syst., Man, Cyber.,* vol. 19, pp. 1426–1445, Nov./Dec. 1989.

[6] W. E. L. Grimson and D. P. Huttenlocker, "On the sensitivity of the Hough transform for object recognition," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 12, pp. 255–274, Mar. 1990.

[7] W. E. L. Grimson and T. Lozano-Perez, "Model-based recognition and localization from sparse range or tactile data," *Int. J. Robotics Res.,* vol. 3, no. 3, pp. 3–35, 1984.

[8] ——, "Localizing overlapping parts by searching the interpretation tree," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. PAMI-9, no. 4, pp. 469–482, 1987.

[9] D. Huttenlocher and S. Ullman, "Recognizing solid objects by alignment with an image," *Int. J. Comput. Vision,* vol. 5, no. 2, pp. 195–212, 1990.

[10] D. W. Thompson and J. L. Mundy, "Three-dimensional model matching from an unconstrained viewpoint," in *Proc. IEEE Int. Conf. Robotics and Automation,* Raleigh, NC, Mar. 1987, pp. 208–220.

[11] W. B. Thompson *et al.,* "Map-based localization: The "drop-off" problem," in *Proc. Image Understanding Workshop, DARPA,* Pittsburgh, PA, Sept. 1990, pp. 706–719.

[12] R. Talluri and J. K. Aggarwal, "Edge visibility regions—A new representation of the environment of a mobile robot," in *IAPR Workshop Machine Vision Applications, MVA '90,* Tokyo, Japan, Nov. 1990, pp. 375–380.

[13] J. J. Koenderik and A. J. V. Doorn, "The internal representation of solid shape with respect to vision," *Biological Cyber.,* vol. 32, pp. 211–216, 1979.

[14] H. Platinga and C. R. Dyer, "Visibility, occlusion, and the aspect graph," *Int. J. Comput. Vision,* vol. 5, pp. 137–160, 1990.

[15] J. H. Stewman and K. W. Bowyer, "Direct construction of the perspective projection aspect graph of convex polyhedra," *Comput. Vision Graphics Image Process.,* vol. 51, pp. 20–37, July 1990.

[16] Z. Gigus, J. Canny, and R. Seidel, "Efficiently computing and representing aspect graphs of polyhedral objects," in *Proc. 2nd Int. Conf. Computer Vision,* Tampa, FL, Dec. 1988, pp. 30–39.

[17] K. Ikeuchi and T. Kanade, "Applying sensor models to automatic generation of object recognition programs," in *Proc. 2nd ICCV,* Tampa, FL, Dec. 1988, pp. 228–237.

[18] K. W. Bowyer, "Aspect graphs: An introduction and survey of recent results," *Int. J. Imaging Syst. Technol.,* vol. 2, pp. 315–328, 1990.

[19] L. Stark and K. Bowyer, "Aspect graphs and nonlinear optimization in 3-d object recognition," in *Proc. 2nd ICCV,* Tampa, FL, Dec. 1988, pp. 501–507.

[20] K. W. Bowyer, "Why aspect graphs are not (yet) practical for computer vision—A discussion," in *IEEE Workshop Directions in Automated CAD-Based Vision,* Maui, HI, June 2–3, 1991, pp. 98–104.

[21] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction.* New York: Springer-Verlag, 1985.

[22] G. T. Toussaint, "Computing the visibility properties of polygons," in *Pattern Recognition and Artificial Intelligence.* Amsterdam: Elsevier, 1988, pp. 103–122.

[23] J. O'Rourke, *Art Gallery Theorems and Algorithms.* New York: Oxford Univ. Press, 1988.

[24] D. H. Green, "The decomposition of polygons into convex parts," *Advances Computing Res.,* vol. 1, pp. 235–259, 1983.

[25] R. Talluri and J. K. Aggarwal, "Positional estimation of a mobile robot using edge visibility regions," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition, CVPR '91,* Maui, HI, June 1991, pp. 714–715.

[26] ——, "Mobile robot self-location using constrained search," in *Proc. IEEE Workshop Intelligent Robots and Systems, IROS '91,* Japan, Nov. 1991.

[27] D. H. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," *Pattern Recognition,* vol. 13, no. 2, pp. 111–122, 1981.

[28] G. Stockman, S. Kopstein, and S. Benett, "Matching images to models for registration and object detection via clustering," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. PAMI-4, pp. 229–241, May 1982.

[29] R. Talluri and J. K. Aggarwal, "Transform clustering for model-image feature correspondence," in *IAPR Workshop Machine Vision Applications, MVA '92,* Tokyo, Japan, Dec. 1992, pp. 579–582.

[30] Y. Liu, T. Huang, and O. D. Faugeras, "Determination of the camera location from 2-d to 3-d line and point correspondences," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 12, no. 1, pp. 28–37, 1990.

[31] R. Talluri and J. K. Aggarwal, "Autonomous navigation in cluttered outdoor environments using geometric visibility constraints," in *Int. Conf. Intelligent Autonomous Systems: IAS-3,* Pittsburgh, PA, Feb. 1993.

[32] B. Nudel, "Consistent labeling problems and their algorithms: Expected-complexities and theory-based heuristics," *Artificial Intell.,* vol. 21, pp. 135–178, 1983.

Raj Talluri (S'89–M'91) received the Bachelor of Engineering degree in electronics and communications engineering from Andhra University, India, in 1984, the Master of Engineering degree in applied electronics from Anna University, India, in 1986, and the Ph.D. degree from the University of Texas at Austin in 1993.

He has been working as a Member Technical Staff at Texas Instruments Corporate Research since 1993. He has also worked as a Scientist at the Indian Space Research Organization's Satellite Center, Bangalore, India, from 1986 to 1987 and as a Graduate Research Assistant at the Computer and Vision Research Center, University of Texas at Austin, from 1988 to 1992.

Dr. Talluri is a member of the IEEE Computer Society. His current research interests include video compression, computer vision, robotics, and computer graphics.

J. K. Aggarwal (S'62–M'65–SM'74–F'76) received the B.S. degree from the University of Bombay in 1957, the B.Eng. degree from the University of Liverpool in 1960, and the M.S. and Ph.D. degrees from the University of Illinois at Urbana in 1961 and 1964, respectively.

He has served on the faculty of the College of Engineering, the University of Texas at Austin, 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 include computer vision, parallel processing of images, and pattern recognition.

Dr. Aggarwal received the Senior Research Award of the American Society of Engineering Education in 1992. He is the author or editor of 7 books and 37 book chapters; and author of more than 150 journal papers, as well as numerous proceedings papers, and technical reports. He has served as Chairman of the IEEE Computer Society Technical Committee on Pattern Analysis and Machine Intelligence (1987–1989); Director of the NATO Advanced Research Workshop on Multisensor Fusion for Computer Vision, Grenoble, France (1989); and Chairman of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (1993). He currently serves as IEEE Computer Society representative to the International Association for Pattern Recognition, is Past President of the International Association for Pattern Recognition, and is an Editor of IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS.