MVA'90 IAPR Workshop on Machine Vision Applications Nov. 28-30,1990, Tokyo

Edge Visibility Regions - A New Representation of the Environment of a Mobile Robot[†]

Raj Talluri and J. K. Aggarwal

Computer and Vision Research Center
The University of Texas at Austin
Austin, Texas 78712

Abstract

This paper proposes a novel representation of the free space of mobile robot by distinct, non-overlapping regions called *Edge Visibility Regions* (EVRs). An algorithm to partition the free space into EVRs from a given stored world model is presented. The EVRs have the property of capturing the geometric relations between the features in the world model with respect to their visibility from different positions in the plane in which the robot navigates. Associated with each region is a list world model features that are visible in that region called the *visibility list* (VL). Also computed and stored for each edge in the VL of an EVR is a range of orientations of the robot for which this edge is visible in the region. An analytic bound on the maximum number of EVRs that will be generated for a given world model of the environment is derived. The advantages of an EVR representation in positional estimation and path-planning tasks of the mobile robot are discussed.

1 Introduction

The problem of navigating a mobile robot autonomously using visual sensors has received considerable interest in the computer vision community. The robot can be 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 a reduced uncertainty, and then allows the robot to perform other navigational tasks. One of the problems in such an approach is that the sensor readings and the world model may be in different formats and co-ordinate frames.

In their work on the PSEIKI system, Kak et al. [4] consider the problem of navigating a mobile robot inside a building using a CAD model of the building and visual sensors. Talluri and Aggarwal [10] consider the positional estimation of an outdoor robot in a mountainous terrain given a Digital Elevation Map (DEM) and a visual camera. Miller [6] and Drumheller [1] use line segments extracted from a sonar sensor as features to match against a map of the environment for positional estimation.

[†] This research was supported in part by NASA under grant NAG 9-361 and Army Research Office under contract DAAL03-87-K-0089.

In this paper, we propose a new intermediary description of the environment to alleviate the problem of matching the sensor observations to the world model. The robot can form this representation offline using the given world model. We consider the case of a mobile robot navigating in a structured, man-made, urban environment consisting of polyhedral buildings. The world model of the robot is assumed to consist of the 3-d descriptions of the rooftops of the buildings. Such a description may be obtained from a pair of stereo aerial images or from the architectural plans of the buildings.

The viewing plane of the robot (the plane in which the robot navigates) is divided into distinct, non-overlapping regions called the *Edge Visibility Regions* (EVRs). These EVRs essentially capture the geometric relations between the model edges with regard to their visibility from various regions in the viewing plane. Associated with each EVR is a *Visibility List* (VL), which is a list of the model edges visible in that EVR; also stored for each edge in the VL of an EVR is the range of orientation angles of the robot for which the edge is visible in this EVR. Thus, each EVR is a region of space which has the topological property that from all the points in it, the same set of edges of the model are visible through a complete circular scan. An upper-bound on the maximum number of EVRs that would be generated for a given world model is derived. The uses of such an EVR representation of the environment in positional estimation, path-planning, and other navigational tasks is also discussed.

2 Edge Visibility Regions

This section presents an algorithm to divide the free space of the robot into the EVRs. The environment of the robot is assumed to consist of polyhedral buildings and the 3-d descriptions of the rooftops, of which are assumed to be given as the world model.

Let OXYZ be the world co-ordinate system (WCS) in which the world model is described. The robot is assumed to be equipped with a camera that can be panned and tilted with no roll. Let O'X'Y'Z' be the robot (camera) centered co-ordinate system (CCS). See Figure 1. In order to estimate the position and pose of the robot in the world co-ordiante system, the transformation matrix $\mathcal{T}$ that transforms WCS into CCS needs to be solved. In general, this matrix has six degrees of freedom: three rotational ψ, ϕ, θ and three translational X, Y, Z, Hence $\mathcal{T}$ is a function of $(X, Y, Z, \psi, \phi, \theta)$. However, in the present problem since it is assumed that 1) the roll angle, $\psi = 0$, 2) the tilt angle ϕ is measurable (a constant), and 3) the

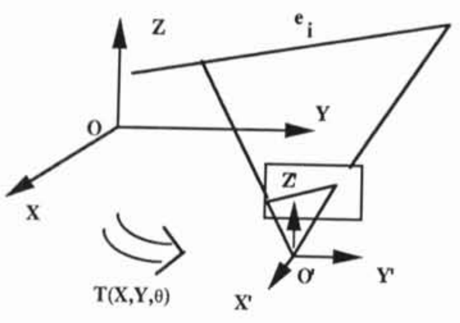

Figure 1: The world and robot co-ordinate systems

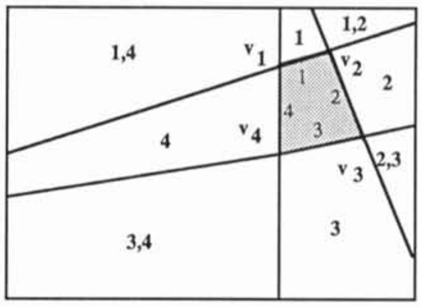

Figure 2: The EVRs after Split

robot is on the ground plane $Z = 0$, there are only three degrees of freedom for this matrix: two translational, X and Y and one rotational, θ. So, the robot navigates in the XY-plane of the world co-ordinate system and has an orientation, θ, which is a negative rotation about the Z-axis. So $T(X, Y, \theta)$ transforms the WCS into the CCS.

Initially, we consider the restrictive case of all the buildings as having flat, convex rooftops, and of being the same height. We then develop an algorithm to generate the EVRs and their associated VLs. The extensions of this algorithm to the general case are then discussed. In the former case, only the projections of the buildings onto the XY-plane need to be considered in forming the EVRs and the VLs, since the tilt angle ϕ is assumed to be measurable.

The problem can now be formally stated as follows:

To partition a plane containing m, n-sided non-intersecting convex polygons into distinct, non-overlapping regions called the *Edge Visibility Regions*(EVRs). These EVRs are characterized by the following properties:

- associated with each EVR is a list of edges of the polygons that are *visible* in that region called as the *visibility list*(VL).
- no two adjacent EVRs have the same VL.

An edge l of a polygon is considered to be visible from a point p if there is at least one point $q \in l$ such that the line segment pq is not intersected by any other edge of any other polygon.

Note that this problem is similar in spirit to the problem of computing the *weak visibility polygon* considered in computational geometry [7,11]. Another problem bearing similarities to this problem is that of forming the aspect graph of a polyhedral object for object recognition [3,9]. However, the above methods concern mainly a single polyhedral object and divide the 3-d view space into regions based on the visibility of faces, edges, vertices, etc. of the object. The present method considers multiple objects and their occlusions. The objects considered are non-overlapping, 2-d, convex polygons, and the viewing space is a 2-d plane.

The algorithm *partition* that divides the XY plane into the desired EVRs, along with their associated VLs, is presented below. The algorithm uses three subprocesses called *split*, *project*, and *merge*. The basic idea of the algorithm 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 EVRs that are intersected are divided into two new ones. The new EVRs then replace the old one and the VLs of the new EVRs are updated to account for the visibility of this edge by considering the edge to be visible in one half-plane, say left of the edge, and invisible in the other. This is handled by the *split* process. For each new polygon considered, the mutual occlusion of the edges of this polygon with the other existing polygons is handled by forming the *shadow region* of these edges on the other existing polygons. This is handled by the *project process*. Finally the *merge* process concatenates all the adjacent EVRs with identical VLs into one EVR. After partitioning the XY-plane into EVRs, for each model edge in the VL of an EVR the range of orientations of the robot for which this model edge is visible in the EVR is also computed and stored. An efficient method to compute this range is also described.

Split

Given a 2-d convex polygon in a plane and a set of existing EVRs, this process is used to update the EVR list to account for the visibility of the edges of the polygon. Consider an n-sided convex polygon P in the viewing plane defined as a collection of n vertices, $v_1, v_2, ... v_n$, and n edges, $v_1v_2, v_2v_3, ... v_{n-1}v_n, v_nv_1$, such that no pair of non-consecutive edges shares a point. Then,

1. extend each of the edges of P, $v_1v_2,...,v_nv_1$, 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 EVRs, 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.

The above procedure also returns the polygon itself as one of the EVRs. This is then removed from the EVR list by checking the list to see if any of the EVRs are the intersecting the polygon. Figure 2 shows the EVRs and their VLs after extending all the edges of P.

Project

This section discusses a process called *project*, which 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 shown in Figure 3. We now need to find the

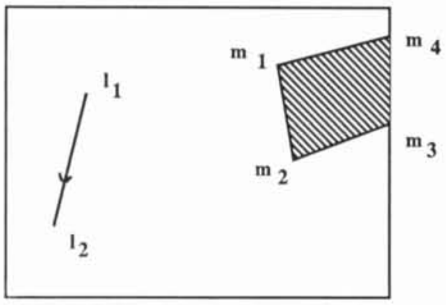

Figure 3: The shadow region for two lines

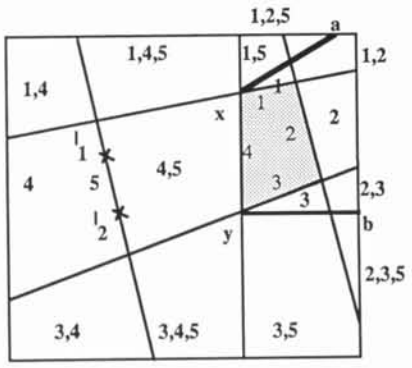

Figure 4: The shadow regions for subcase 1

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 union 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. There are two sub cases to be considered:

- subcase 1 : In this subcase 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. Figure 4 shows this.

- subcase 2 : In this subcase, only part of the polygon lies to the left of the line. That is, the extension of the line cuts some of the edges of the polygon. Figure 5 shows this.

The determination of the shadow vertices is different in each of these subcases. So, the process *project* detects these subcases and uses a different method in each. For subcase 1, shown in Figure 4, the shadow vertices are found as follows: the vertex of P corresponding to the max of the angle $\angle l_2l_1v_i$, $i = 1,2,....n$, gives the shadow vertex x, and similarly the vertex of P corresponding to the maximum of $\angle l_1l_2v_i$, $i = 1,2,....n$, gives the shadow vertex y.

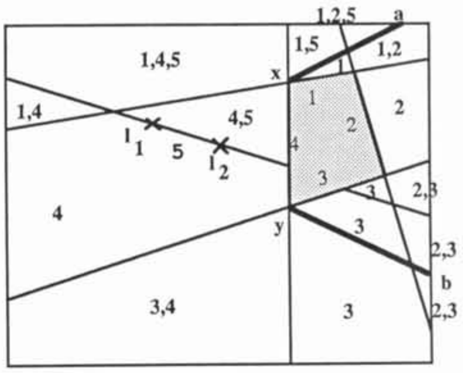

Figure 5: The shadow regions for subcase 2

For subcase 2, shown in Figure 5, it is slightly more involved to find x and y. First, the convex hull of the points $v_1, v_2, v_3, ...v_n$ and l_2 is formed. Then the two vertices of the convex hull adjacent to l_2, v_1 & v_3, are considered. Of these, the vertex to the left of the line l_1l_2 is considered to be the shadow vertex y and the other to be x. Hence, $x = v_1$ and $y = v_4$. To find the shadow region, the line l_1x is extended and its intersection with the plane is found as a; similarly 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 the Figure 5, is marked as invisible and, hence, removed from the VLs of all the EVRs 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 EVRs with identical VLs. However, the *Merge* process, to be discussed next, accounts for this.

Merge

Given an EVR list and the associated VLs, this process is used to search the list for adjacent EVRs that have identical VLs. Since the EVRs are actually convex polygons, two EVRs are considered as adjacent if they have one edge common. These EVRs are then merged into a single EVR, and the EVR list is updated if they have identical VLs. This step is run iteratively, each time merging the adjacent EVRs and forming new ones until no two adjacent EVRs have identical VLs.

2.1 The Algorithm *Partition*

The algorithm *partition* is described below, which given a list of convex polygons and a viewing plane, divides the viewing plane into EVRs and forms their associated VLs. The algorithm initially assumes the entire viewing plane to be one EVR and then iteratively splits this plane into sub-regions, considering each of the polygons in turn using the process *Split*. Whenever a new polygon is considered, not only are the existing EVRs split to account for the visibility of its edges, but the occlusions of the edges of this polygon 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 EVRs that have the same VLs. As described before, the routine *Split(P, List)*, takes a polygon P and an EVR list *List*, and modifies the EVRs in *List* and their associated VLs 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 VLs to take into account the occlusion of e due to the edges of P. The *Merge(List)* routine takes an EVR list, *List*, merges the adjacent EVRs having the same VLs, and returns the modified *List*.

Algorithm Partition
Input : A set of convex polygons in a plane.
Output : A set of EVRs and their associated VLs.

```
1. Initialize
    1.1 Initialize the $EVR\_LIST$ to contain one region,
                                  the entire 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
```

Figure 6 shows the EVRs and their VLs for the case of two convex polygons.

2.2 The Estimation of the range of orientations

For each model edge in the VL of an EVR the range of orientations of the robot for which this model edge is visible in the EVR is also computed and stored. An efficient

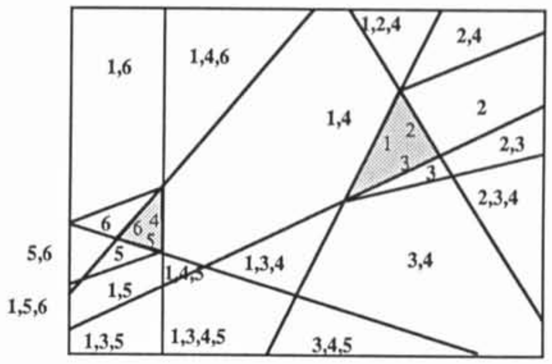

Figure 6: The EVRs of two convex poygons

method to compute the range is given below. Given an EVR and a model edge e_i in the VL of the EVR, we need to find the lower bound, θ_{min}, and the upper bound, θ_{max}, of the orientation angles of the robot (camera) for which this edge e_i is visible in the EVR. Let the vertices of the EVR be $v_i, i = 1, ..n$. Let p_1, p_2 be the end points of e_i. Then it can be easily shown that θ_{min} is the minimum of the angles made by the lines joining p_2 to $v_i (i = 1, ..n)$ and that θ_{max} is the maximum of the lines joining p_1 to $v_i, i = 1, ..n$. Hence, given an EVR and a model edge e_i in its VL, the maximum and the minimum orientation angles can be estimated by considering just the vertices of the EVR.

2.3 Extensions

In the case of the buildings with rooftops that are not convex, the non-convex polygons representing the projections of the rooftops on to the XY-plane are decomposed into a set of adjacent, component convex polygons [5,7]. Only decompositions without *Steiner points* are considered, since the modifications to the existing algorithms in this case are straightforward. The extra edges that are addded in the process are considered as *dummy* edges, and their visibilty is not marked in the VLs of the EVRs. Hence, the self occlusions of the edges of a non-convex polygon are handled by decomposing the polygon into component convex polygons and dealing with their mutual occlusions by using the *project* process. In the case when all the buildings are not of the same height, it is insufficient to just consider the projections of the rooftops onto the XY-plane alone when forming the shadow regions. The project process is modified to consider a Z-shadow line also. Figure 7 illustrates this case. Here the shadow region of line a on to the polygon P is, hence, the region bounded by L, M and N, where N is the Z-shadow line. Note that the case of all the buildings of the same height is actually a special case of unequal height buildings where the Z-shadow line is at infinity. In the case when the rooftop of a building is not flat (planar), it is decomposed into convex planar polygons and each of these is considered as a separate polygon; the partition algorithm is then modified as before, in the non-convex case. Also, the Z-shadow lines are drawn to account for these component polygons, which are now convex and flat but of different heights.

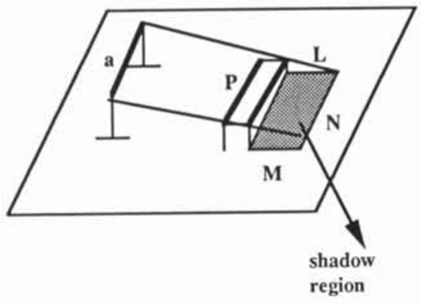

Figure 7: The shadow regions for buildings of unequal height

3 Estimating the Number of EVRs

An interesting and important question related to using this method is how many EVRs would be generated using the *partition* algorithm. If this number is very large, it would be impractical to use the method. One might think that the number of distinct EVRs grows exponentially with m and n the number of polygons and the number of sides of each polygon, respectively. We derive an upper bound on the maximum number of EVRs that would be generated and show that this 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 [3] in the context of aspect graphs for object recognition.

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

$$\begin{aligned} L(k) &= L(k-1)+n+2\cdot n\cdot(k-1) \\ &+ 2\cdot n\cdot(k-1) \qquad (1)\end{aligned}$$

where the first term indicates the number of lines already existing when $(k-1)$ polygons have been considered; the second term indicates the number of lines drawn by extending the edges of the *kth* polygon; 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 edge of a polygon contributes two *shadow lines* for each of the existing polygons lying on the visible side of the edge. Solving (1) we have,

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

Let $T(m)$ denote the maximum number of EVRs generated when we have m n-sided, convex polygons in a plane under perspective projection. Note that $T(0)=1$. Consider the situation when there are $m-1$ polygons already existing and 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 which increase the number of lines and hence the EVRs. 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 EVRs grows after each new line is added. We can derive the maximum number of EVRs by induction. Let $T(m)_k$ denote the number of EVRs after the kth line is drawn. Since we started with $T(m-1)$ EVRs and $L(m-1)$ lines, after the first additional line is drawn, the number of EVRs, $T(m)_1$, 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 will divide $L(m-1)+1$ regions into two, thus adding $L(m-1)+1$ new EVRs. Proceeding in this fashion,

$$\begin{aligned} T(m)_2 &= T(m)_1+L(m-1)+2 \\ T(m)_{n(4m-3)} &= T(m)=T(m)_{n(4m-3)-1} \\ &+ L(m-1)+n(4m-3) \\ T(m) &= T(m-1)+n(4m-3)L(m-1) \\ &+ \sum_{i=1}^{n(4m-3)} i \\ &= T(m-1)+\frac{[n(4m-3)]^2}{2}+\frac{n(4m-3)}{2} \\ &+ n(4m-3)[n(m-1)(2m-3)] \\ &= T(m-1)+O(n^2m^3)+O(n^2m^2) \\ &+ O(mn) \\ &= O(n^2m^4) \end{aligned}$$

It is worth pointing out that although the maximum number of regions suggested by this derivation is of $O(n^2m^4)$, in practice it is very unlikely that so many EVRs 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 to be drawn to all the existing polygons at any given time. In practice, however, we need to draw the shadow lines only onto the polygons that lie on the visible side of the edge. Since we are considering convex polygons, the situation of all the polygons lying on the visible side of all the edges is an impossible case. Also, the *Merge* process is quite effective and reduces the number of regions by a significant amount, particularly as more more and more polygons are considered. Taking all these factors into account, 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 EVRs is, in the worst case, still polynomial in n and m.

4 Positional Estimation and Path Planning

Having developed a methodology to partition the free space in the into EVRs, we now propose a procedure to estimate the position and pose of the robot in the environment. Also, we discuss the use of the EVRs in path-planning is discussed.

4.1 Positional Estimation

As discussed in section 2, the position of the robot is given by three parameters, (X, Y, θ). These can be obtained by solving the transformation matrix $\mathcal{T}(X, Y, \theta)$, which, transforms the WCS into the CCS. This matrix can be solved for by establishing a correspondence between the features in the WCS and the CCS. The features we plan to use are the line segments that constitute the roof tops of the buildings, since the 3-d descriptions of these are assumed to be given in the world model.

To establish the correspondence between the world model features and their images, one possibility is to formulate the problem as a search paradigm [2], i.e., to form an *interpretation tree* of all the possible pairings between the model and the image features and to search this interpretation tree for a set of consistent pairings which can then be used to solve the transform $T(X, Y, \theta)$. However, since the 3-D description of the model features are given, by using the geometric relations between them, and the known perspective geometry of the camera, we can formulate constraints as to which features will be visible from different places in the XY-plane. The interpretation tree can then be pruned using these constraints so as not to consider all the possible pairings. Recall that the EVRs store all the relevant geometric information about the model features, like, the number of features visible from a given region and the range of orientations of the robot for which these features are visible. So given a set of images features the EVRs can be used to prune the interpretation tree in establishing a consistent set of pairings between the image and model features and, hence, the position and pose of the robot can be computed accurately from this set of matches.

Usuallly the robot position and pose, as given by its position encoders, is available, which essentially isolates the robot position to lie within a few EVRs. So only these EVRs need to be considered in searching for the exact location Also if the robot can identify a landmark (one of the 3-D features), this ability can be used to isolate the robot position to lie only within those EVRs that contain this landmark edge in their VLs. Similarly, if a rough estimate of the θ value is available, it can be used to quickly prune some of the hypotheses.

4.2 Path Planning

Planning paths from the start to the goal and executing them is an important navigational task to be performed by all autonomous mobile robots [8]. The EVRs are a convex region representation of the free space. Representing the free space by convex regions has the advantage that any path chosen inside an EVR will be entirely within the EVR and, hence, will be obstacle free. So, to plan an obstacle free path between a start position S and a goal position G, the free space is represented as a graph, with the EVRs as the nodes and an arc existing between two EVRs, if they are adjacent. If S is in EVR_i and G is in EVR_j, then an obstacle free path between S and G can be found by a graph search for a connected component list between EVR_i and EVR_j. If the arcs are weighted by the distance metric or some other function to be optimized, the graph search can be used to yield the minimum distance path. Also, since each EVR has a VL of the world model edges, this can be used to establish the robot's position in each EVR accurately at every desired instant and, hence, the path can be followed more precisely.

5 Conclusions

In this paper we present a novel representaion of the environmet of a mobile robot called the *Edge Visibility Regions*. This representation can be formed offline from a given stored model description of the environment. The uses of representing the environment as EVRs in alleviating the problem of establishig a correspondence between the sensor readings and the stored model for the various naviagtional tasks of the mobile robot like positional estimation and path planning are discussed. The ideas presented in this paper are currently being implemented using a polygonal model of an urban scene.

References

[1] M.Drumheller, "Mobile robot localization using sonar," *IEEE Tran. on PAMI*, 9(2), March 1987, pp. 325-332.

[2] 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.

[3] K.Ikeuchi and T.Kanade, "Applying sensor models to automatic generation of object recognition programs," in *Proceedings of 2nd ICCV*, pp. 228-237, 1988.

[4] A.Kak, K.Andress and C.Lopez-Abadia, "Mobile robot self-location with the PSEIKI system," *Working Notes, AAAI spring symposium on robot navigation*, Mar 1989, pp.33-37.

[5] J.M.Keil and J.R.Sack, " Minimum decomposition of polygonal objects", *Computational Geometry*, edited by G.T. Toussaint, Elsevier Science Publishers B.V., pp. 197-216, 1985.

[6] D.Miller, "A spatial representation system for mobile robots," *Proceedings of the IEEE Int. Conf. Robotics and Automation*, pp. 122 - 127, St.Louis,1985.

[7] Joseph O'Rourke, *Art Gallery Theorems and Algorithms*, John Wiley, 1988.

[8] J.T.Schwartz, M.Sharir and J.Hopcroft, *Planning, Geometry, and Complexity of Robot Motion*, Ablex publishing co., N.J., 1987.

[9] L.Stark and K.Bowyer, "Aspect graphs and non-linear optimization in 3-D object recognition," in *Proceedings of 2nd ICCV*, pp. 501-507, 1988.

[10] R.Talluri and J.K.Aggarwal, "Position estimation techniques for a mobile robot in an unstructured environment," in *Proc. IROS '90*, pp..189-166, Japan, July 1990.

[11] G.T.Toussaint, "Computing visibility properties of polygons," *Pattern Recognition and Artificial Intelligence*, Elsevier Science Publishers B.V., pp. 103-122, 1988.