A Dynamic Visibility Inference Scheme
Based on New Spatial Knowledge Representations from Observer's Perspective

Yeong-Ren Jean and Hsing-Pang Chang

Department of Computer Science and Information Management,
Providence University, Taichung, Taiwan, R.O.C.
yrjean@pu.edu.tw and garros@mail.ltu.edu.tw

ABSTRACT

In this article, we propose two new spatial knowledge representations called OG-string and Visibility-string for a dynamic environment seen from the observer's perspective. The OG-string preserves the objects' VBSs (Virtual Blocking Set) and can be used to determine visibility of objects. Because of having the same VBS, objects with the same begin-bound in ring-direction are grouped together. As a result the size of OG-string also can be reduced. The Visibility-string contains the objects' visibility information, and thus can cooperate with OG-string to provide enough spatial information for dynamic visibility inference. We also represent a Dynamic Visibility Inference Scheme consisting of an Initialization, an Addition and a Deletion Algorithm based on the two new representations. The scheme assist with visibility inference for a dynamic environment where objects are added and deleted frequently just like in a warehouse so that a mobile robot's path planning can be improved.

1: INTRODUCTIONS

In recent years, applications involving digital images and multimedia are increasing in number day by day. To make better use of these images, the spatial knowledge representation that is used to describe the spatial relations between objects in the images is being innovated continually. One of the applications is the visibility inference that the visibility of objects seen from a point of view of an observer can be inferred. Just like a robot viewing the goods from a point in a warehouse. In the past, many methods to represent the spatial relations between objects have been presented. Some are based on a Cartesian coordinate system [1-6] and the basic concept is to project the objects depicted in a picture along x- and y-coordinates and to capture two strings to represent the relative positions of objects in the x- and y-axis. Alternatively, some are based on a polar coordinate system [7-10] that projects all objects along radial and angular coordinates. The ring- and sector-directions associated with these coordinates used to create string again representing relative positions of objects. In polar coordinates, the PCOS-string [11] is found to be particularly useful for visibility inference.

2: VISIBILITY INFERENCE BASED ON PCOS-STRING

In this section, we discuss Huang's [11] visibility inference algorithm which uses a PCOS-string for inferring the visibility of objects from the observer's perspective. Assume that there is an image with several objects and we can trace two bounds for each object along the ring-direction or sector-directions based on a polar coordinate system [10]. In the ring-direction, each of the two bounds is the point of tangency and both of them are generated by concentric circles outward along the ring-direction from the viewing point of an observer. The point closer to the observer is called the begin-bound while the farther point is called the end-bound. In the sector-direction, the two bounds are determined by a half-line that rotates in clockwise direction. The begin-bound and end-bound are the first and second point of tangency found for each object encountered during a complete revolution. As we show in Figure 1, b^r_T and e^r_T are the begin- and end-bound of O_T in the ring–direction and b^s_T and e^s_T are the begin- and end-bound of O_T in the sector–direction. Therefore the spatial position of O_T can be specified as (b^r_T , e^r_T , b^s_T , e^s_T). If there are n objects in the picture then we can generate a PCOS-string containing the spatial positions of all n objects. The PCOS-string is of the form{O_1(b^r_1 , e^r_1 , b^s_1 , e^s_1), O_2(b^r_2 , e^r_2 , b^s_2 , e^s_2) …O_n(b^r_n , e^r_n , b^s_n , e^s_n)} with $b^r_1 \le b^r_2 \le \ldots \le b^r_n$. For example, the PCOS-string for the picture of the Figure 2 is presented as {O_1(11, 66, 1, 3), O_2(22, 55, 7, 8), O_3(33, 77, 2, 5), O_4(44, 99, 9, 10), O_T(88, 110, 4, 6)}.

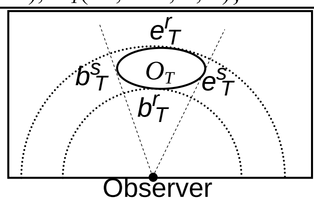

Figure1. The spatial position of O_T can be specified as (b^r_T , e^r_T , b^s_T , e^s_T).

To understand how the visibility inference algorithm handles its task using a PCOS-string, we first introduce some concepts about VBS (Virtual Blocking Set). It is significant that each object in a PCOS-string is projected into the ring-direction and sector-direction and that there are two projection intervals that can be found for each object, one for ring-direction and one for sector-direction. Of course, we can discover two projection intervals in the ring-direction or sector-direction associated with two nonzero sized objects and find out the spatial relationships of the two intervals through the spatial knowledge representation. There are 13 possible spatial relationships [12] of the two objects in one direction shown in Figure 3. All of the spatial relationships can be represented by the seven spatial operators [5] whose notations and semantics are given in Table 1. If and only if the intersection of the above two projection intervals is not empty, we say that the two intervals can be **merged** in the sector-direction. Then, the interval $\{\min(b^s_1, b^s_2)\ \max(e^s_1, e^s_2)\}$ is called the connected projection interval merged from the object O_1 and the object O_2 in the sector-direction. A MCPI (Maximal Connected Projection Interval) is an interval such that no other projection intervals in the same direction can be **merged** with it. The VBS (Virtual Blocking Set) for a given object O_T is the set of all MCPIs in the sector-direction that were merged from the projection intervals along the sector-direction for all objects preceding the object O_T. The arrangement of objects illustrated in Figure 2 provides a setting in which to determine an example of VBS for the object O_T. The VBS is the set of intervals {(1,5)(7,8)(9,10)} and can determine the visibility of the object O_T.

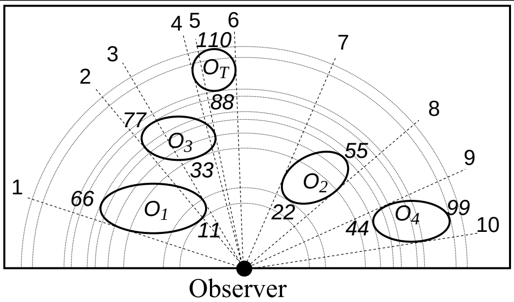

Figure 2. The VBS for O_T is the set of {(1,5)(7,8)(9,10)}.

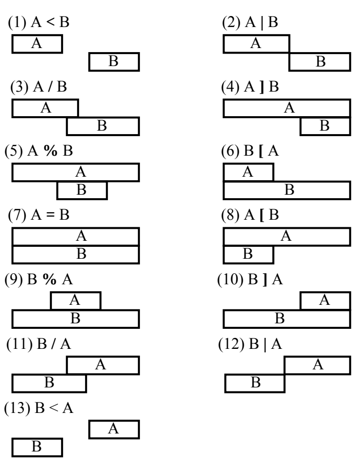

Figure 3. The 13 possible spatial relationships between any two objects in one direction.

The VBSs of all objects' in the image can be computed using the PCOS-string and can be used to infer all objects' visibility through a process that compares every object's projection interval in sector-direction with its VBS, separately. Based on those concepts, Huang proposed a visibility inference algorithm [11] that can handle both static and dynamic situations. However, the algorithm is complex because every object's VBS must be re-computed for dynamic visibility inference.

Notations	Conditions
A < *B*	end(*A*) < begin(*B*)
A = *B*	begin(*A*) = begin(*B*) , end(*A*) = end(*B*)
A | *B*	end(*A*) = begin(*B*)
A % *B*	begin(*A*) < begin(*B*) , end(*A*) > end(*B*)
A [*B*	begin(*A*) = begin(*B*) , end(*A*) > end(*B*)
A] *B*	begin(*A*) < begin(*B*) , end(*A*) = end(*B*)
A / *B*	begin(*A*) < begin(*B*) < end(*A*) < end(*B*)

Table 1. The definitions of spatial operators.

Assume that there are *n* objects in a dynamic environment and some object is added or deleted. Then a DS (Derivative Scope) will be produced by a deleted object or a DB (Derivative Blocking) produced by an added object. As seen in Figure 4, if the object O_2 is the deleted/added object and then the DS/DB will be the interval ([5,10]). In fact, the DS/DB is also the VS (Viewed Scope) of the object. Moreover, some objects may be viewed by the DS for deletion or some objects may be obstructed by the DB for addition. An easy explanation is as follows: objects preceding the object in the ring-direction can be treated as the preceding objects, and objects succeeding the object in the ring-direction can be treated as the succeeding objects. If the object is invisible as the object O_3 in Figure 4 then we can say that the preceding and succeeding objects won't be affected because the DS/DB of O_3 is empty. If the object is partly- or fully-visible such as object O_6 in Figure 4, we can say that preceding objects (O_1, O_2, O_3, O_4 and O_5) won't be affected. And say that some succeeding objects (O_9, O_{10} and O_{11}) won't be affected because the DS/DB won't affect those objects' visibility and some other succeeding objects will be affected (O_7 and O_8) because those objects can be viewed/obstructed by the DS/DB.

A more efficient means of dynamic visibility inference requires objects' viewed state and visible scope which forming our Visibility-string and objects' VBSs which forming our OG-string. The OG-string and Visibility-string facilitate inference of objects' visibility without re-computing and re-inferring wholly.

3: NEW SPATIAL KNOWLEDGE REPRESENTATIONS

Two new spatial knowledge representations, called OG-string and Visibility-string, can be used to speed up the dynamic visibility inference. Assume that there are *n* objects in the image field and they are grouped according to their begin-bound in ring-direction. Because of having the same VBS, objects with the same begin-bound in ring-direction are grouped together. Thus, their visibility can be determined by the VBS called GVBS (Group's Virtual Blocking Set). This kind of group can be formed as $\{G_j\,(C_j, Gb^r_j, GVBS_j)\}$, where the G_j is the jth group and the C_j is a counter to record how many objects there are in the group G_j. The Gb^r_j is equal to each object's begin-bounds in ring-direction in the jth group. The $GVBS_j$ is equal to each object's VBS in the jth group. Then the OG-string can be formed as $\{ G_1 (C_1, Gb^r_1, GVBS_1), G_2 (C_2, Gb^r_2, GVBS_2) \ldots G_j (C_j, Gb^r_j, GVBS_j) \ldots G_m(C_m, Gb^r_m, GVBS_m) \}$ with $1 \leq m \leq n$. If all n objects have the same begin-bound in ring-direction, m=1. If all n objects have different begin-bound in ring-direction, m=n.

The Visibility-string contains the viewed state and visible scope of all objects in the image field. It is of the form $\{V_1 (S_1, VS_1), V_2 (S_2, VS_2) \ldots V_i (S_i, VS_i) \ldots V_n (S_n, VS_n)\}$, where the V_i is the visibility information of the ith object. S_i is the ith object's viewed state. Each object's viewed state can be invisible (IV), partly-visible (PV) or fully-visible (FV). The visible scope of the ith object is VS_i. If the S_i is IV, VS_i is equal to ø. If the S_i is PV then VS_i is of the form ([p_1], [p_2] …[p_x]) for x visible pieces where each piece is represented in the form [begin-bound, end-bound]. Finally, the VS_i is equal to ([b^s_i, e^s_i]) if the S_i is FV. For example, the object O_5 in Figure 4, has a visibility information $V_5(S_5, VS_5)$ given by V_5(PV,[4,5][10,11]).

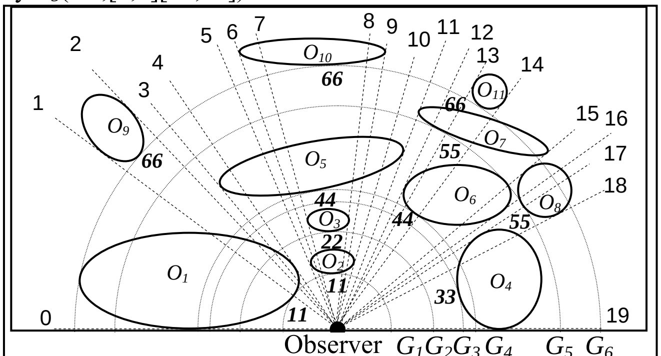

Figure 4.

1. The OG-string is{G_1(2,11,(ø)),G_2(1,22([0,4][5,10])), G_3(1,33,([0,4][5,10])),G_4(2,44,([0,4][5,10][17,19])) , G_5(2,55,([0,11][12,16][17,19])),G_6(3,66,([0,19]))}
2. The Visibility-string is{V_1(FV,([0,4])),V_2(FV,([5,10])), V_3(IV,(ø)),V_4(FV,([17,19])), V_5(PV,([4,5][10,11])), V_6(FV,([12,16])),V_7(PV,([11,12])) ,V_8(PV,([16,17])) , V_9(IV,(ø)) , V_{10}(IV,(ø)) , V_{11}(IV,(ø))}.

Figure 4 provides an example of the OG-string and Visibility-string for an entire field of objects. Objects O_5 and O_6 have the same begin-bound (44) in the ring-direction and of course share the same GVBS ([0,4][5,10][17,19]) and thus they are grouped into the same group G_4(2,44,([0,4][5,10][17,19])). To compare their projection intervals in sector-direction (O_5=[3,11], O_6=[12,16]), traced from PCOS-string, with the sharing GVBS respectively will produce their visibility information {V_5(PV,([4,5][10,11])), V_6(FV,([12,16]))}.

4: A DYNAMIC VISIBILITY INFERENCE SCHEME

The Dynamic Visibility Inference Scheme consists of three primary algorithms; Initialization, Addition and Deletion Algorithm, which use PCOS-string, OG-string and Visibility-string to assist with dynamic visibility inference. There are two primary situations, addition and deletion, in the dynamic environment. For the Deletion Algorithm, we define three Interval Difference Operations, denoted by ⊟, ⊟ and ⊟. They can be used for two overlapping intervals to execute the difference operation which compares a projection interval with one of the pieces of the DS. Each operation provides difference functionality and they are detailed in Figure 5. Besides, an Interval Mergence Operation, denoted by ⊞, is used for the Addition Algorithm. It deals with the situations where two intervals are adjacent or overlapping as seen in Figure 6. After the Mergence Operation for two intervals, another interval is produced by a combination of the two intervals.

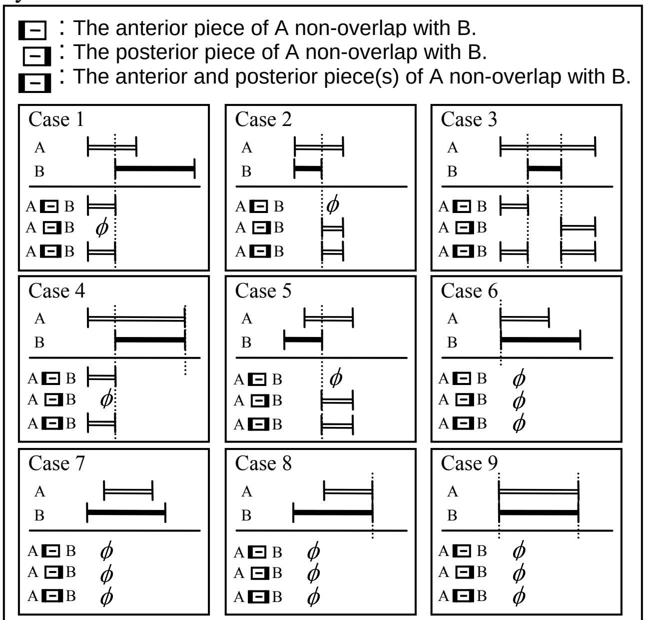

Figure 5: The all possible cases of Interval Difference Operations, denoted by ⊟, ⊟, and ⊟.

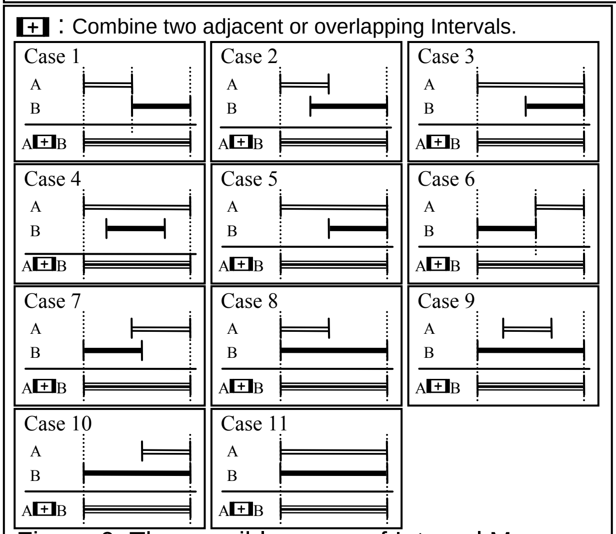

Figure 6: The possible cases of Interval Mergence Operation, denoted by ⊞.

The previously described Visibility Inference Algorithm, based on the PCOS-string, was intended to infer given objects' visibility and output a visibility list. This algorithm will become the Initialization Algorithm in our improved dynamic visibility inference scheme. The task of the former algorithm was only to infer the visibility of objects and output a visibility list. The task of the new, modified, Initialization Algorithm is to gather the visibility and VBS information for all objects in the field and be able to output the OG-string and Visibility-string for additions and deletions associated with further dynamic change. The pseudo-code for the Initialization Algorithm is shown in Figure 7. We also use the I^s_i to indicate the projection interval of the object O_i in the sector-direction. The related subroutines are shown in Figure 8 and are also used in the Deletion Algorithm and Addition Algorithm too.

Initialization Algorithm
Input:PCOS-string
Outputs:OG-string, Visibility-string
1. $i = 0$, $j = 0$, $M = \phi$, $Gb^r = \phi$, Visibility-string= ϕ ,
    OG-string= ϕ
2. Find the next m objects { $O_{i+1}, O_{i+2} \ldots O_{i+m}$ } with $b^r_{i+1} = b^r_{i+2} = \ldots = b^r_{i+m}$.
3. j++ , $k = 1$
4. $GVBS_j = M$; $G_j = [\, m, b^r_{i+m}, GVBS_j]$
5. OG-string= OG-string $\cup$ G_j
6. while($k \le m$){
    if there exists a $MCPI \in GVBS_j$ such that
    ($MCPI = I^s_{i+k}$) or ($MCPI$ [I^s_{i+k}) or
    ($MCPI$] I^s_{i+k}) or ($MCPI$ % I^s_{i+k})
    {
        S_{i+k} = IV // S_i is invisible.
    } else if there exists a $MCPI \in GVBS_j$ such that
    ($MCPI$ / I^s_{i+k}) or (I^s_{i+k} / $MCPI$) or
    (I^s_{i+k} [$MCPI$) or (I^s_{i+k}] $MCPI$) or
    (I^s_{i+k} % $MCPI$)
    {
        S_{i+k} = PV // S_i is partly-visible.
    } else {
        S_{i+k} = FV // S_i is fully-visible
    }
    VS_{i+k} = AnalyzeObjectVisibleScope(I^s_{i+k} , $GVBS_j$)
    $V = V_{i+k}$ (S_{i+k} , VS_{i+k})
    Visibility-string= Visibility-string $\cup$ V
    M = AdjustGVBSbyMergence(I^s_{i+k} , M)
    k ++
    }
7. if ($i + m == n$){
        Return the OG-string and Visibility-string
    }else{ $i = i + m$, GoTo 2 }

Figure 7. Initialization Algorithm.

It is important that the Visibility-string stores not only viewed state but also visible scope, denoted by S and VS. This information is necessary for the Deletion Algorithm. As pointed out above, VS is also a DS when an object is deleted. As a result, in the case of dynamic deletion, it is easy to determine whether an object's viewed state, S, is IV, PV or FV, and an object's visible scope, VS, is empty, one piece or several pieces. Assume that the deleted object is the ith object and resides in jth group in OG-string. If the S_i is IV then OG-string and Visibility-string can be immediately refined without re-computing and re-inferring. It is only necessary to remove the deleted object's related information from the original PCOS-string and Visibility-string. Moreover, it is needed to remove G_j from the original OG-string if the C_j equals to 1 or to subtract 1 from the C_j if the C_j is greater than 1.

Subroutine: AnalyzeObjectVisibleScope (I, $GVBS$)
Inputs: I, $GVBS$
Output: VS
1. k=1, $VS = \phi$, $temp = I$
2. y=the number of the MCPIs in $GVBS$
3. while($k \le y$ && $temp \ne \phi$){
    if($temp$ overlaps $MCPI_k$){
        $VS = VS \cup (temp \boxminus MCPI_k)$
        $temp = temp \boxminus MCPI_k$
    }
    k ++
}
4. $VS = VS \cup temp$
5. return VS

Subroutine: AdjustGVBSbyMergence (I, $GVBS$)
Inputs: I, $GVBS$
Output: $GVBS_{new}$
1. $temp = I$, $i = 1$
2. y = the number of the MCPIs in the $GVBS$
3. while($i \le y$){
    if($temp \ne \phi$){
        if there exists a spatial relationship such that ($MCPI_i < temp$){
            $GVBS_{new} = GVBS_{new} \cup MCPI_i$
        }else if there exists a spatial relationship such that ($temp < MCPI_i$){
            $GVBS_{new} = GVBS_{new} \cup temp \cup MCPI_i$
            $temp = \phi$
        }else{ $temp = temp \boxplus MCPI_i$ }
    }else{
        $GVBS_{new} = GVBS_{new} \cup MCPI_i$
    }
    i ++
}
4. if(y ==0){ return ($GVBS_{new} \cup temp$) } else { return $GVBS_{new}$}

Subroutine: AdjustDSorDB (I, DS(or DB))
Inputs: I , DS(or DB)
Output: DS_{new}(or DB_{new})
1. $k = 1$; $DS_{new} = \phi$
2. z = The number of the derivative scope (or blocking) pieces.
3. while($k \le z$){
        if(I overlaps p_k) { $p_k = (p_k \boxminus I)$ }
        $DS_{new} = DS_{new} \cup p_k$
        k++
    }
4. return DS_{new}

Subroutine: GetFirst**Object**IndexInAdjustingArea (O,PCOS-string)
inputs: PCOS-string, O //The object can be specified as (b^r_T, e^r_T, b^s_T, e^s_T).
output: i
1. $i = 1$, n = the number of PCOS-string
2. while($i \le n$){
        if($b^r_i > b^r_T$){ break }
        i++
    }
3. return i

Subroutine: GetFirst**Group**IndexInAdjustingArea (O,OG-string)
inputs: OG-string, O //The object can be specified as (b^r_T, e^r_T, b^s_T, e^s_T).
output: j
1. $j = 1$, m=the number of OG-string
2. while($j \le m$){
        if($Gb^r_j > b^r_T$){ break }
        j++
    }
3. return j

Subroutine: AdjustGVBS(DS , $GVBS$)
Inputs: DS , $GVBS$

Figure 8. Subroutines.

```
Output: GVBS
1. i = 1, k =1, temp = ϕ , GVBS_new = ϕ , result = ϕ ;
2. y = The number of the MCPIs in the GVBS.
3. z = The number of the pieces in the DS.
4. while(i ≤ y){
   if(MCPI_i overlap p_1){
      temp = MCPI_i
      while(k ≤ z && temp≠ ϕ ){
         if( temp overlaps p_k ){
            result = result ∪ (temp ⊟ p_k)
            temp = temp ⊟ p_k
         }
         k++
      }
      result = result ∪ temp
      replace MCPI_i with result
   }
      GVBS_new = GVBS_new ∪ MCPI_i
      i++
}
5. return GVBS_new
```

Figure 8. Subroutines (continued).

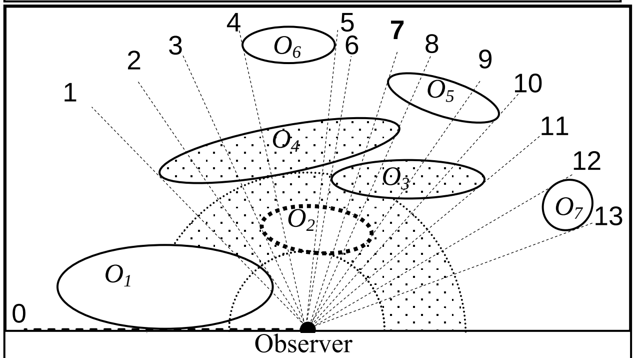

Figure 9. The Adjusting Area is denoted by dotes and consists of object O_3 and O_4.

On the other hand, if the status of the deleted object O_i is PV or FV, it is also unnecessary to re-compute all objects' VBSs and re-infer all objects' visibility information. All objects reside in one of three areas, the Anterior Area, the Adjusting Area or the Posterior Area. Objects in the Anterior Area share a characteristic that their begin-bounds are less than the begin-bound of deleted object in ring-direction and it means that those objects precede the deleted object in ring-direction. Therefore, we can confirm that their VSs and VBSs are not be effected by the DS. The objects that are in the Adjusting Area share the feature that the objects' visibility are affected by the DS because initially they are partly or fully obstructed by the deleted object. But now, they may partially or fully exposed by the DS. Finally, if the DS is entirely obstructed by some object along ring-direction, we can call the object a critical object. It signifies that the affection of the DS disappears due to the critical object. Objects following the critical object will belong to the Posterior Area. In the other words, the visibility of the objects in Anterior Area and Posterior Area will not be affected. More precisely, the Adjusting Area is the area where objects succeed the deleted object and precede the critical object. AS a result it is only necessary to re-compute the VBSs and re-infer the visibility information of the objects that belong to the Adjusting Area. Figure 9 illustrates what happens when the object O_2 is deleted. The Adjusting Area is denoted by the dotted region in Figure 9 and includes the objects O_3 and O_4. The DS ([3,9]) is produced by the deletion of the object O_2 and disappears behind the critical object O_4.

Without loss of generality, the DS may have several pieces in sector-direction and each piece can be formed as [begin-bound, end-bound]. The DS can be of the form ($[p_1][p_2]\ldots[p_z]$) if it contains z pieces. The Deletion Algorithm is shown in Figure 10.

```
Deletion Algorithm
Inputs:PCOS-string , Visibility-string , OG-string , O_d //O_d is the
   deleted object.
Outputs: PCOS-string , Visibility-string , OG-string
1. DS = VS_d
2. if(S_d ≠ " IV " ){
   i = GetFirstObjectIndexInAdjustingArea(O_d , PCOS-string)
   g = j = GetFirstGroupIndexInAdjustingArea(O_d , OG-string) -1
   /* re-compute objects' GVBSs and re-infer objects' visibility
information in Adjusting Area */
   while(i ≤ n and DS ≠ ϕ ){
         if(Gb^r_j ≠ b^r_i ){
            j++
            GVBS_j = AdjustGVBS(DS, GVBS_j)
         }
         if there exists a MCPI ∈ GVBS_j such that
         ( MCPI = I^s_i ) or ( MCPI [ I^s_i ) or
         ( MCPI ] I^s_i ) or ( MCPI % I^s_i )
         {
            V_i = (IV, ϕ )        // invisible
         } else if there exists a MCPI ∈ GVBS_j such that
         ( MCPI / I^s_i ) or ( I^s_i / MCPI ) or
         ( I^s_i [ MCPI ) or ( I^s_i ] MCPI ) or
         ( I^s_i % MCPI )
         {
            VS_i=AnalyzeObjectVisibleScope(I^s_i , GVBS_j)
            V_i = (PV , VS_i)     // partly-visible
         } else {
            V_i = (FV , I^s_i )   // fully-visible
         }
         if( I^s_i overlaps DS ){
            DS = AdjustDSorDB ( I^s_i , DS )   // Decline the DS.
         }
         i++
   }
}
3. Delete the O_d from PCOS-string and the V_d from Visibility-string.
4. if( C_g+1==1 ){ Delete the G_g+1 from OG-string }else{ C_g+1 -- }
5. Return PCOS-string, Visibility-string and OG-string
```

Figure 10. Deletion Algorithm.

Just as a DS can be generated by a deleted object, a DB can be generated by an added object. The DB can be trace from a process that to compare the added object projection interval of sector-direction with some GVBS. The GVBS reside in one of the objects' group in OG-string that is the first group in the Adjusting Area or the group and added object with the same begin-bound in ring-direction. The process of visibility inference for addition is similar to the process of visibility inference for deletion and the Adjusting Area also can be found. Moreover, each object's VBS and visibility in the Adjusting Area also can be re-computed and re-inferred. The process will go on until the DB declines and disappears or no more any projection interval can be compared with DB along ring-direction. It should also be noted that the DB are of the form ($[p_1][p_2]\ldots[p_z]$) if there are z pieces. Each piece also can be formed as [begin-bound, end-bound]. The Addition Algorithm is presented in Figure 11.

Addition Algorithm
Inputs: PCOS-string , Visibility-string , OG-string and O_a // O_a is the added object.
Outputs: PCOS-string , Visibility-string , OG-string
1. $G_{new} = \phi$
2. i = GetFirst**Object**IndexInAdjustingArea (O_a , PCOS-string)
3. j = GetFirst**Group**IndexInAdjustingArea (O_a , OG-string) -1
4. if($Gb^r_j == b^r_a$){
    $G^r_j = (C_j + 1, Gb^r_j , GVBS_j)$
    $DB = VS_a$ = AnalyzeObjectVisibleScope (I^s_a , $GVBS_j$)
  }else{
    $Gb^r_{new} = b^r_a$
    $GVBS_{new} = GVBS_{j+1}$
    $G_{new} = (1, Gb^r_{new}, GVBS_{new})$
    $\mathbf{DB} = VS_a$ = AnalyzeObjectVisibleScope(I^s_a , $GVBS_{new}$)
  }
5. if ($VS_a == I^s_a$) { S_a = FV } // fully-visible
  else if ($VS_a == \phi$) { S_a = IV } // invisible
  else { S_a = PV } // partly-visible
6. $V_a = (S_a, VS_a)$
7. if($S_a \neq$ " IV "){
/* re-compute objects' GVBSs and re-infer objects' visibility information in Adjusting Area */
  while($i \leq n$ && $DB \neq \phi$){
    if($Gb^r_j \neq b^r_i$){
      j++
      $GVBS_j$ = AdjustGVBSbyMergence(I^s_a, $GVBS_j$)
    }
    if there exists a $MCPI \in GVBS_j$ such that
    ($MCPI = I^s_i$) or ($MCPI$ [I^s_i) or
    ($MCPI$] I^s_i) or ($MCPI$ % I^s_i)
    {
      $V_i = (\mathrm{IV}, \phi)$ // invisible
    } else if there exists a $MCPI \in GVBS_j$ such that
    ($MCPI$ / I^s_i) or (I^s_i / $MCPI$) or
    (I^s_i [$MCPI$) or (I^s_i] $MCPI$)or
    (I^s_i % $MCPI$)
    {
      VS_i = AnalyzeObjectVisibleScope (I^s_i ,$GVBS_j$)
      $V_i = (\mathrm{PV}, VS_i)$ // partly-visible
    } else {
      $V_i = (\mathrm{FV}, I^s_i)$ // fully-visible
    }
    if(I^s_i overlaps DB){
      $\mathbf{DB}$ = AdjustDSorDB (I^s_i , DB) // Decline the DB
    }
    i++
  }
}
8. Insert the O_a into PCOS-string and the V_a into Visibility-string.
9.if ($G_{new} \neq \phi$){ Insert G_{new} into OG-string }

Figure 11. Addition Algorithm.

5: CONCLUSION

A visibility inference mechanism with high performance is essential for a dynamic environment where objects are added or deleted frequently. In this article, two new spatial representations are proposed called OG-string and Visibility-string for all objects in a dynamic environment seen from the observer's perspective. The OG-string contains the GVBSs of all objects with the same begin-bound in ring-direction. This grouping helps to reduce the size of OG-string. Moreover, all objects' visibility can be determined through the OG-string. The Visibility-string contain all objects' visibility information and thus can cooperate with the OG-string and PCOS-string to provide enough spatial information including each object's VBS, visibility state, visible scope and projection interval for dynamic visibility inference. We also propose a Dynamic Visibility Inference Scheme consisting of three algorithms based on PCOS-string, OG-string and Visibility-string for the dynamic environment seen from the observer's perspective without the need to re-compute and re-infer all objects' VBSs and visibility information. The dynamic visibility inference algorithm provided by this scheme gives a robust and efficient method to assist in guiding a mobile robot's path as it navigates through a changing warehouse environment.

6: REFERENCES

[1] S.K. Chang, Q.Y. Shi, and C.W. Yan, ''Iconic indexing by 2-D strings,'' *IEEE Trans. Patt. Anal. Mach. Intell.,* PAMI-9**,** 413–428 (May 1987).

[2] S.Y. Lee, M.C. Yang, and J.W. Chaen, ''2D B-string: a spatial Knowledge Representation for Image Database Systems,'' *proc. ICSC'92 Second Int. Computer Sci. Conf.*, pp. 609-615, 1992.

[3] E. Jungert, ''Extended Symbolic Projection used in a knowledge structure for spatial reasoning,'' *The 4th BPRA Conference on Pattern Recognition.* Springer V, Cambridge, pp.343-351, 1988.

[4] S.K. Chang, E. Jungert and Y.Li, ''Representation and Retrieval of Symbolic Pictures using Generalized 2D String,'' *Technical Report, University of Pittsburh*, 1988.

[5] S.Y. Lee and F.J. Hsu, ''2D C-string: A New Spatial Knowledge Representation for Image Database Systems,'' *Pattern Recognition*, vol. 23 no. 10, pp. 1077-1088, 1990.

[6] S.Y. Lee and F.J. Hsu, ''Spatial Reasoning and Similarity Retrieval of Images Using 2D C-string Knowledge Representation,'' *Pattern Recognition*, vol. 25, no. 3, pp. 305-318, 1992.

[7] G. Petraglia, M. Sebillo, M. Tucci, and G. Tortora, ''Towards Normalized Iconic Indexing,'' *Proc. 1993 IEEE Symp. Visual Language*, pp. 392-394, 1993.

[8] G. Petraglia, M. Sebillo, M. Tucci, and G. Tortora, ''Rotation-invariant Iconic Indexing for Image Database Retrieval, Progress in Image Analysis and Processing III,'' S. Impedovo, ed., pp. 271-278, World Scientific, 1994.

[9] G. Petraglia, M. Sebillo, M. Tucci, and G. Tortora, Senior Member, ''Virtual Images for Similarity Retrieval in Image Databases,'' *IEEE Transactions Knowledge and Data Engineering*, vol. 13 no. 6, pp. 951-967, 2001.

[10] P.W. Huang and Y.R. Jean, ''Spatial Reasoning and Similarity Retrieval for Image Database Systems Based on RS-string,'' *Pattern Recognition*, vol. 29 no. 12, pp. 2103-2114, 1996.

[11] P. W. Huang, P. L. Lin, "Visibility Inference Based on Spatial Knowledge Representation From Observer's Perspective," *International Journal of Intelligent Systems*, Vol. 12, pp. 191-202, 1997.

[12] J.F. Allen, ''Maintaining knowledge about temporal intervals,'' *Commun. ACM*, vol. 26 no. 11, 832–843, 1983.