

- [153]Y. Chen and G. Medioni, "Object Modelling by Registration of Multiple Range Images," *International Journal of Image and Vision Computing (IVC)*, 10(3):145–155, April 1992.
- [154]C.-K. R. Chung and R. Nevatia. Recovering building structures from stereo. In *Proceedings of the IEEE Workshop on Applications of Computer Vision*, Palm Springs, CA, December 1992.
- [155]D. Kim and R. Nevatia. Indoor navigation without a specific map. *Intelligent Autonomous Systems*, 1993.
- [156]D. J. Kriegman, E. Triendl, and T. O. Binford. Stereo vision and navigation in buildings for mobile robots. *IEEE Trans. on Robotics and Automation*, 5(6), December 1989.
- [157]E. Le Bras-Mehlman, M. Schmitt, O. D. Faugeras, and J. D. Boissonnant. How the delaunay triangulation can be used for representing stereo data. *Procs. of Int. Conf. on Computer Vision*, 1988.
- [158]R. Mohan and R. Nevatia. Using Perceptual Organization to Extract 3-D Structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1121–1139, November 1989.
- [159]H. P. Moravec. *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*. PhD thesis, Stanford University, Stanford, California, September 1980. Technical Report AIM-340 and STAN-CS-80-813.
- [160]M. H. Soldo. Reactive and preplanned control in a mobile robot. *Procs. of Image Understanding Workshop*, 1990.

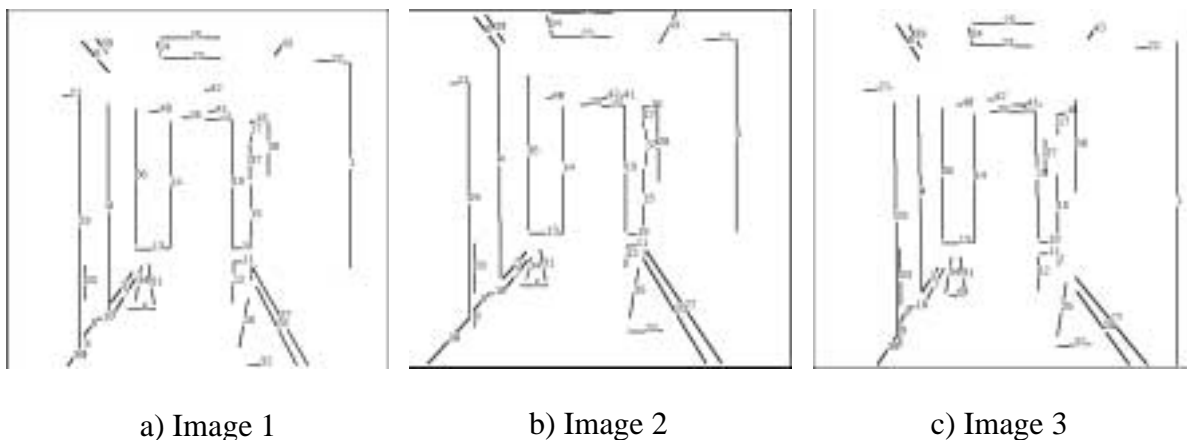


Figure 9.13 Matched segments for Corridor

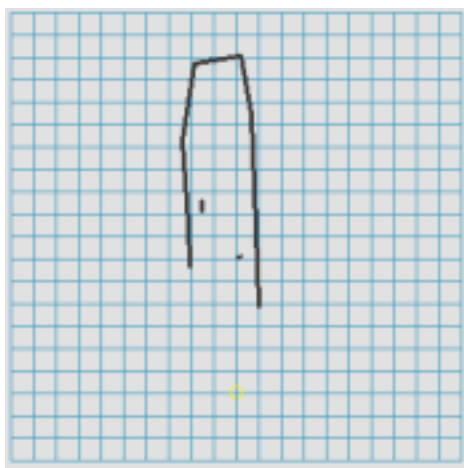
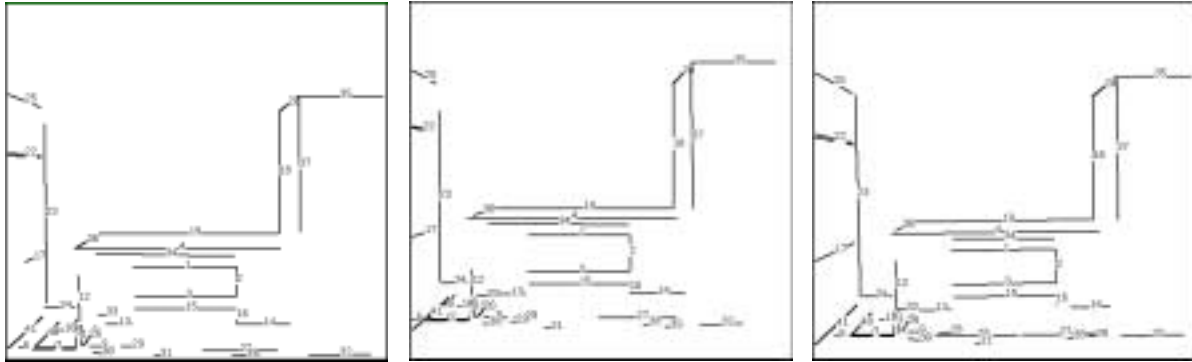


Figure 9.14 S-map for Corridor

- [148]N. Ayache and F. Lustman. Fast and reliable passive trinocular stereovision. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 422–427, London, England, June 1987.
- [149]J.D. Boissonnat, O.D. Faugeras, and E. L. Brass-Mehlman. Representing stereo data with the delaunay triangulation. *IEEE Int. Conf. on Robotics and Automation*, 1988.
- [150]D. Braunegg. Marvel: A system that recognizes world locations with stereo. *Procs. of Int. Conf. on Computer Vision and Pattern Recognition*, 1991.
- [151]E. Bruzzone, M. Cazzanti, L. De Floriani, and F. Mangili. Applying two-dimensional delaunay triangulation to stereo data interpolation. *European Conference on Computer Vision*, 1992.
- [152]J. F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, November 1986.



a) Image 1

b) Image 2

c) Image 3

Figure 9.10 Matched segments for Laboratory 2



Figure 9.11 S-map for laboratory 2



a) Image 1

b) Image 2

c) Image 3

Figure 9.12 Corridor Scene



a) Image 1

b) Image 2

c) Image 3

Figure 9.9 Laboratory 2

Figure 9.12 shows an image frame for a corridor. Moreover, Figure 9.13 displays matched segments with the s-map given in Figure 9.14. Two walls of the corridor and one wall at the end of the corridor are represented. Moreover, two traffic cones are successfully represented.

9.5 Conclusion

In this paper, we introduced a new representation of an indoor environment, s-map, which can represent the location of obstacles in a planar map. The obstacles are defined as any objects that can block robot movement. The s-map utilizes the viewing triangle constraint and the stability constraint. These constraints provide both efficient validation of the vertical surface hypothesized by two adjacent vertical lines and efficient grouping of the partial vertical surface having a single vertical line. The s-map is made by mapping the 3-D segments of smaller objects and verified vertical surfaces into a planar map. The mapping is done by dropping height information of the 3-D segments or the vertical surfaces. Therefore, this mapping allows the s-map to be made directly from 3-D segments without reconstructing a 3-D open space and to represent obstacles in a planar map that is a navigable map for the robot moving in a plane. In addition to the efficient map making, the s-map transforms many 3-D problems in navigation into 2-D problems because it represents objects in 2-D domain. However, the s-map is applicable only in limited environments because it assumes flat horizontal floors and vertical walls, and it degrades when there are wrong matches becoming inhibition segments. Our future effort will be directed toward extending the s-map for more complex environments and adding robustness in validating vertical surfaces against wrong matches becoming inhibition segments.

9.6 References for Chapter

[147]N. Ayache. *Artificial Vision for Mobile Robots*. The MIT Press, 1991.

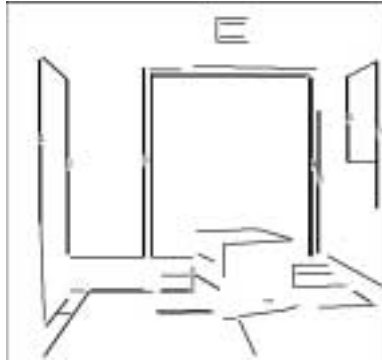


Figure 9.7 Vertical segments of Laboratory 1

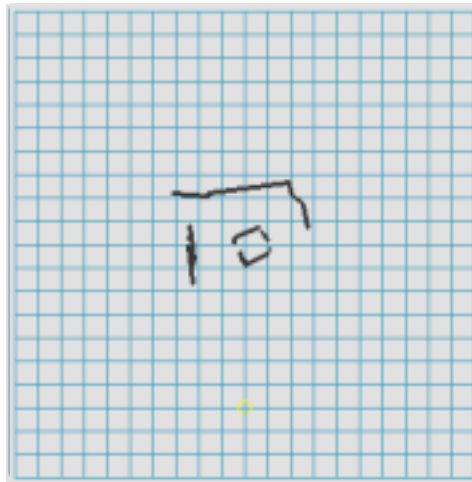


Figure 9.8 S-map of Laboratory 1

the vertical segment 7 is a ghost vertical segment made by the support of the vertical segment 6. Then the verification and the mapping are done. The locations of two walls, a box, and a cabinet are successfully represented in the s-map. Moreover, the right wall is extended with the vertical and its support: Segment 1 and Segment 2 in Figure 9.1 . Similarly, the left wall that is between the cabinet and the door, is also extended with the vertical segment and its support: Segment 12 and Segment 10 in Figure 9.1 .

Figure 9.9 shows an image frame for a laboratory. Moreover, Figure 9.10 displays matched segments of Figure 9.9 . The s-map is given in Figure 9.11 . The locations of a desk, an air conditioner, and a cabinet are represented in the s-map. A wall behind the desk is also represented in the s-map. Moreover, front part of the cabinet is extended with the vertical segment and its support: Segment 37 and Segment 35 in Figure 9.10 . Similarly, the air conditioner is also extended with the vertical segment and its support: Segment 23 and Segment 22 in Figure 9.10 .

Finally, both mislocated matches and wrong matches show the same error pattern because both of them generate a 3-D segment having wrong 3-D information. They cause more damage to taller objects if they are vertical lines of the taller surfaces. They shift location of the vertical surfaces. However, for smaller objects, they do not affect the location of the object. They just place themselves in wrong locations because the s-map does not make surfaces by combining boundary lines of smaller objects, and may prevent two adjacent vertical segments from making a vertical surface when the wrong matches become inhibition segments of them.

9.3.3 Advantages and disadvantages of the s-map

The advantages of the s-map are fourfold: Making the map is very simple. The map is directly made from 3-D segments, so it is not necessary to reconstruct a 3-D open space. The second advantage is that the s-map itself is a navigable map. Therefore, we do not have to cut a 3-D space to find a navigable map. In conventional methods [157,151], it is very difficult to find a cutting height level of a 3-D space for navigation. Since a reconstructed 3-D space is produced by visible line segments and the lower part of objects may be invisible, the reconstructed 3-D space could be too conservative at the lower level of a navigation environment so that only small free area is available for navigation. Thus, finding a cutting level is a compromise between area of navigable space and safety. Therefore, deciding the cutting level is difficult. Thirdly, the s-map represents the environment more realistically and completely with relatively few 3-D segments. This comes from the utmost use of characteristics of indoor environments, which are embedded in the viewing triangle constraint and the stability constraint. Finally, the s-map can turn many 3-D problems in navigation into 2-D problems, for example, path planning and map fusion. Therefore, many navigation problems can be solved efficiently. The disadvantage of the s-map is that wrong matches may prevent two adjacent vertical segments from making a vertical surface when the wrong matches become inhibition segments.

9.4 Results

We have tested the s-map in indoor environments, such as corridors and laboratories. Moreover, our robot has successfully navigated corridors in our building by representing the corridors with the s-map [155].

In this paper, we present the s-maps for two laboratory image frames and one corridor image frame, which are illustrated in Figure 9.8, Figure 9.11, and Figure 9.14. As we can see in these s-maps, the accuracy of the s-maps depends on the accuracy of the reconstructed 3-D segments. Therefore, a more accurate s-map can be acquired with more accurate the reconstructed 3-D segments. This depends on accuracy of a stereo system.

Figure 9.1 shows an image frame for a laboratory. Moreover, Figure 9.1 displays matched segments of the Figure 9.1. Now vertical segments are selected. Figure 9.7 illustrates the selected vertical segments. Among the vertical segments,

environments. Thus we estimate the complexity to be $O(n)$ in real cases though a much deeper analysis is necessary. Finally, the complexity of the verification algorithm is the same as that of distribution. The complexity of s-map is $O(n^2)$ in worst case, and $O(n)$ in real cases.

The run time of the making the s-map from reconstructed 3-D segments was about one tenth of a second in our experiments. The algorithm was run in the Sun Sparcstation 10. The algorithm is currently programmed in Lisp. Moreover, the current program is not an optimized one. Thus the run time can be reduced by optimizing it. In addition, the run time can be further reduced by mutiprocessors because each slice can be verified independent of the other slices.

9.3.2 Reliability of the s-map

Graceful degradation is one of the important aspects for vision algorithms because vision algorithms deal with real pictures whose quality varies according to environmental changes. For the s-map algorithm, there are two places where errors come in: Edge detection and matching. In edge detection, the most common errors are missing edges, shortened or broken edges, and mislocated edges. These errors affect the performance of not only matching but also the s-map algorithm. In matching, common errors are wrong matches, missing matches, and shortened matches. From these two levels of errors, the s-map algorithm suffers from four kinds of errors: partial matches, missing matches, mislocated matches, and wrong matches. For these four cases, the reliability of the algorithm is explored.

In the first case, partial matches do not affect taller object surfaces, but shrink smaller object surfaces when the partial matches are segments of top surfaces. In reconstructing a vertical surface, the necessary information constitutes locations of vertical lines and coplanar segments, as described in section 9.2.4. Therefore, partial matches do not degrade the algorithm of reconstructing vertical surfaces. In reconstructing the smaller objects, only segments of top surfaces are necessary. Thus only partial matches of top surfaces shrink smaller objects. The shrunk surfaces may be extended by grouping the top surfaces. However, such extensions are not necessary in most navigation problems because the shrunk portion of an object is relatively small and does not cause problems for a robot to plan a path with the shrunk object.

In the second case, missing matches can degrade the s-map algorithm more severely. For simple analysis, we assume that every surface is a quadrilateral. For taller objects, even two missing matches are tolerable if they are nonvertical lines. In the case of missing either of the two vertical lines, the shrunk vertical surface can be reconstructed if there is a support. For smaller objects, missing those matches that are not top surface boundaries does not degrade the algorithm of the s-map at all because top boundaries represent the location of the objects in the s-map. Missing some of top boundaries means that an object is shrunk. Even in the case where only one of the top boundaries is detected, the location of the object can be still known.

1) Segmentation of an image

- a) We collect vertical lines taller than the robot.
- b) The image is segmented into vertical slices according to the vertical lines.
- c) Other segments are distributed among the vertical slices. The segments in a slice are categorized into the three groups: coplanar, inhibition, and irrelevant groups.

2) Verification and mapping

- a) In each slice, the validity of the hypothesized vertical surface is checked in terms of the viewing triangle constraint.

If the vertical surface is valid one

then the surface is squeezed into the s-map.

else a support for a vertical segment is searched for.

If there is a support

and the surface made by the support

and the vertical is a valid surface

then the surface is squeezed into

the s-map.

- b) While the validity of a vertical surface is decided, the segments in its slice are squeezed into an s-map if they can block robot movement.

Figure 9.5 Algorithm for building a s-map

9.3 Discussion of the s-map

9.3.1 Complexity of the s-map

The s-map algorithm is analyzed for its complexity. Let's assume that there are n matched segments in an image. Selection of vertical lines takes $O(n)$, and dividing the image needs sorting of the vertical lines. Let the number of vertical lines l and the number of columns c . Because vertical lines have the same column coordinate, the vertical lines can be sorted by a bucket sort method where buckets are image columns. Thus the complexity of sorting is $O(\max(l, c))$. Now nonvertical segments are distributed among slices. They are distributed to those slices that the nonvertical segments straddle. These slices are simply the slices lying between the column coordinates of their end points. The worst case is that all nonvertical segments straddle all vertical segments, where the complexity is $O(l \times (n-l))$. The maximum of this complexity occurs when l is $n/2$. Therefore, the worst case complexity is $O(n^2)$. In real cases, most nonvertical segments straddle less than three vertical slices and the number of vertical slices is less than ten because the number of walls and tall objects is small in indoor

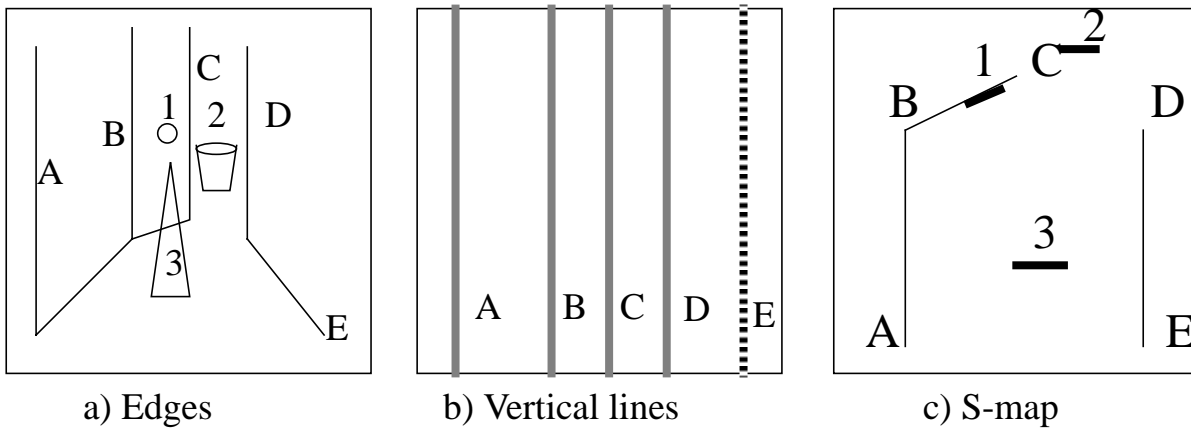


Figure 9.6 Examples of making an s-map

vertical surface is hypothesized with the support and the vertical segment. The hypothesized vertical surface is verified using the viewing triangle constraint, that is, if there is no inhibition segment for the surface hypothesized, the surface is presumed existing.

While verifying the presence of a vertical surface, the segments smaller than the robot are mapped to an s-map by simply dropping height information because the robot can see the boundaries of the top surfaces as described in section 9.2.2.

One example of making an s-map is shown in Figure 9.6 -a, b, and c. One edge image for an indoor scene is given in Figure 9.6 -a. In this figure, we assume that A, B, C, and D are vertical lines and that segments group 1, 2, and 3 are coplanar segments, inhibition segments, and irrelevant segments, respectively. First, this image is segmented. From this image, we can make vertical slices according to vertical lines A, B, C, D, and E. E is a ghost vertical line. Figure 9.6 -b shows the vertical lines. Now other segments are distributed among vertical slices. Segment groups 1 and 3 are distributed into the slice made with B and C, and segment group 2 is distributed into the slice made with C and D. While distributing the segments, we classify the segments into one of three groups: coplanar, inhibition, and irrelevant groups. At the second step, verification is done. The slice made from A and B reconstructs a vertical surface because there are no inhibition segments, and then the vertical surface is mapped to an s-map. Similarly, the slice made from B and C is mapped to the s-map while segments group 3 is mapped to the s-map because the segment group 3 can be an obstacle for the robot. However, the slice made from C and D cannot form a vertical surface because there are inhibition segments that constitute segment group 2. Instead, the segment group 2 is mapped to the s-map because it can block the movement of a robot. The slice made from D and E is mapped to the s-map because there are no inhibition segments. The s-map of Figure 9.6 -a is shown in Figure 9.6 -c.

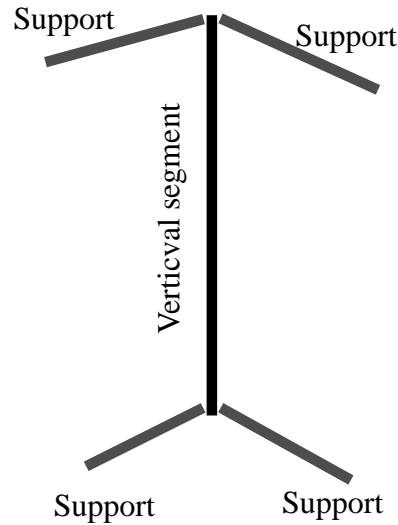


Figure 9.4 The supports for a vertical segments

made by a vertical line and its ghost vertical line does the same verification as ordinary vertical slices do.

9.2.5 Algorithm for making the s-map

The algorithm for making the s-map is in two steps: segmentation of an image into vertical slices, and verification of vertical surfaces for the vertical slices while mapping smaller objects into the s-map. The first step is further divided into three modules: collecting vertical segments taller than a robot, dividing the image among vertical slices according to the collected vertical segments, and distributing other segments among the vertical slices. The algorithm for making a s-map is summarized in Figure 9.5 . The details of the algorithm are given below.

In the first step, we first collect vertical segments including ghost vertical lines. Next, we divide an image into vertical slices according to the vertical segments. Finally, the remaining segments, such as nonvertical segments and the vertical segments smaller than a robot, are distributed among the vertical slices. While being distributed, the segments are categorized into three groups: inhibition segments, coplanar segments, and irrelevant segments. Each segment is distributed among those vertical slices that lie in between the begin and end points of the segment.

At the second step of making an s-map, we verify the presence of the vertical surface hypothesized by a vertical slice while mapping possible obstacles into the s-map. If there are no inhibition segments, a vertical surface is presumed existing. Otherwise, a support for a vertical segment is searched for. In the case, where no support for the vertical segment is found, no further attempt is made to find a partial vertical surface. In the other case, where a support for a vertical segment is found, a partial

triangle constraint generated from the segments between the two adjacent vertical lines.

9.2.4 Stability constraint

We introduce perceptual grouping to extend surfaces that are missed by the reconstruction using the viewing triangle constraints. Although much research has been done on grouping for surface reconstruction both in 2-D and 3-D, the research has tried to group surfaces by hypothesize-and-test methods with geometric properties. One drawback of this approach is that it requires a lot of computation. Therefore, it is not practical for robot navigation where an important concern is real time navigation. We try to achieve speed by searching small areas with the help of the stability constraint. Before explaining grouping, we describe the stability constraint. Then we describe the cases when perceptual grouping is necessary, and the perceptual grouping to extend surfaces.

The stability constraint is described as follows.

Stability constraint *Every object must be stable with respect to gravity.*

Pin-like objects, such as vertical segments, tend to fall down unless there are some supports. Conversely, if there is a vertical segment, then there are supports for it. The supports can be any segments adjacent to the vertical segment, which can keep the vertical segment vertical. In an indoor environment, the vertical segments are assumed to be side ends of vertical surfaces. Thus the supports are the segments that are adjacent to the end points of the vertical segments and are either on a stable plane or beneath a stable plane. Figure 9.4 illustrates the supports for a vertical segment.

Grouping segments of the same vertical surface is necessary when one of our assumptions is loosely preserved. A basic assumption of reconstructing a vertical surface is that both vertical end lines for the surface are visible, but that assumption is not always true. The missing end line occurs in cases of clipping or occlusion. The clipping case occurs when a wall is so large that entire wall is not covered in an image. In the second case, a wall is occluded by other vertical objects, such as a cabinet, so that vertical lines of other objects locate between the two vertical lines of the wall. These two cases cause missing vertical surfaces.

The missing surfaces can be extended if there is a support. When there is more than one support, the farthest support line is selected among them. The support and the vertical line can make a vertical surface if the surface hypothesized by them does not violate the viewing triangle constraint. For easy implementation of the verification, a ghost vertical line is introduced. The ghost vertical line for a vertical line is an imaginary vertical line that is located at the end point of a support line of the vertical line. The ghost vertical line for the clipping case is made when vertical segments are collected, while the ghost vertical line for the occlusion case is made when a vertical slice has inhibition segments. After finding a ghost vertical line, the vertical slice

Observation 1 : Viewing Triangle Constraint *The space confined within a viewing triangle is free space.*

This viewing triangle constraint is used to verify whether two adjacent vertical lines are on the same surface and make a vertical surface. Before explaining how to use the viewing triangle constraint for verification, we describe how to find adjacent vertical lines. First, a vertical line is found in a 2-D image, and the verticality of the vertical line is checked with its 3-D information. Moreover, The vertical line should be also taller than a robot because they are assumed to be boundary vertical segments of vertical surfaces that are taller than a robot. Then, for the vertical line, an adjacent vertical line, which also satisfies the above conditions, is searched for in the image.

After finding two adjacent vertical lines, we collect segments between them. Collecting such segments is done by simply checking column coordinates of segments. Because a vertical segment has the same row coordinate value along the column coordinate, its column coordinate can represent its location in the image. Therefore, segments are collected into a vertical slice made by the two adjacent vertical lines if column coordinates of the segments are located between column coordinates of the two adjacent vertical segments. The segments are then categorized into three groups: coplanar, inhibition, and irrelevant segments.

- coplanar segments : The segments that are coplanar with the hypothesized vertical surface made with two adjacent vertical lines.
- inhibition segments : The segments that are either behind or across the hypothesized vertical surface
- irrelevant segments : The segments that are in front of the hypothesized vertical surface.

When the presence of the surface is verified, the verification is done conservatively to prevent the robot from hitting obstacles. When the robot misses a real wall, it collides with it. Making an imaginary wall only reduces free space, and the reduced free space can be expanded by the following image sequence. Therefore, we presume that there is a wall unless there is evidence of nonexistence of the wall. The evidence comes from the segments in its vertical slice.

The viewing triangle constraint generated by coplanar segments helps two adjacent vertical lines make a vertical surface. However, the viewing triangle constraint generated by inhibition segments prevents the two adjacent vertical lines from making a vertical surface. As the name suggests, irrelevant segments do not affect the making of a vertical surface with the two adjacent vertical lines. Thus verification of presence of a vertical surface for two adjacent vertical lines is done by checking if there are inhibition segments between the two adjacent vertical lines. This verification using inhibition segments leads the following observation.

Observation 2. Verification with viewing triangle constraint *Two adjacent vertical lines can make a vertical surface if the surface does not violate the viewing*

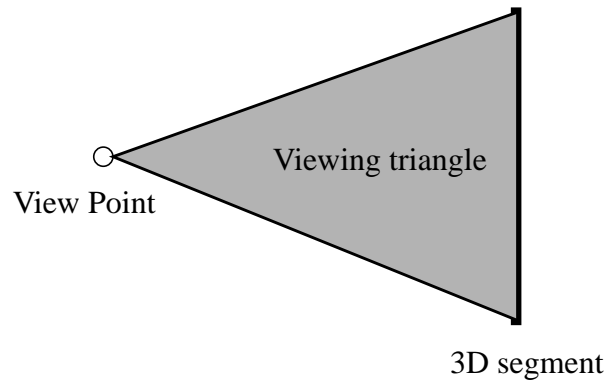


Figure 9.3 Viewing Triangle

tions of the taller objects including walls. If we follow the same process as we do for the smaller objects, we get only points because the vertical lines are points in the 2-D space, which consists of width and length but not of height. Therefore, we need some kind of surface reconstruction with the vertical lines.

The surface reconstruction with vertical lines uses the characteristic of a vertical surface, that is, the vertical surface has two adjacent vertical lines bounding it. However, all adjacent vertical lines are not in the same surface. Two adjacent vertical lines fall into two categories: both of them are on the same surface or they are on different surfaces. Those two adjacent vertical lines that are on the same surface can make a surface. However, those two adjacent vertical lines that are on the different surfaces cannot make a surface. Therefore, vertical surfaces are reconstructed only from the two adjacent vertical lines on the same surface. Then the reconstructed vertical surfaces are squeezed to a floor. Now the problem of surface reconstruction with the vertical lines becomes how to verify if two adjacent lines are on the same surface. The verification process is described in section 9.2.3 .

9.2.3 Viewing triangle constraint

We assume that objects are opaque. Seeing an object means that the visible part of the object is not occluded by any other obstacles. Similarly, a 3-D segment corresponding to the edge detected in an image, is not occluded by any other objects. From the above observation, we introduce the viewing triangle constraint described below. Before explaining the viewing triangle constraint, we define the viewing triangle as follows.

Definition: Viewing triangle *The triangle made with a view point and the 3-D segment corresponding to the edge detected in an image*

Figure 9.3 shows a viewing triangle..

Now the viewing triangle constraint is stated as follows.

planar domain. The term, s-map, comes from the idea that the map is made by squeezing 3-D segments into a 2-D map. In this section, we will first give the definition of the s-map, and exploit characteristics of an indoor environment. Then we will introduce the viewing triangle constraint and the stability constraint, which incorporate the characteristics of an indoor environment to verify vertical surfaces efficiently. We will finally explain how to build the s-map.

9.2.1 Definition of the s-map

The s-map is defined as a map that represents the locations of the visible 3-D surfaces of obstacles in a 2-D space, where 2-D consists of width and length coordinates but does not include a height coordinate. Obstacles are defined as objects that can block the movement of the robot. Therefore, the objects beneath the ceiling are not considered obstacles because they do not block robot movement. Conceptually the s-map is made by first reconstructing a 3-D map and cutting the 3-D map from floor level to robot height level, and finally squeezing the cut 3-D map. However, we do not build the 3-D map, but rather make the s-map directly from 3-D segments.

9.2.2 Characteristics of an indoor environment

The characteristics of an indoor environment are different from those of an outdoor one. Therefore, the characteristics should influence the methodology in order to make the utmost use of them. Now we present the characteristics of indoor environments, and make reasonable assumptions on a navigation environment on which to base the algorithm.

Rooms are mainly composed of vertical walls and horizontal floors, and are filled mostly with man-made objects. Among those man made objects, most objects taller than humans, such as bookshelves, are composed of vertical surfaces. On the other hand, top surfaces of the objects smaller than humans, such as a tea table, can be seen if they are not occluded. From these characteristics, we make three reasonable assumptions for the indoor environment: the floor is flat, and the objects that are taller than a robot, including walls, have vertical surfaces. Moreover, top surfaces of the smaller objects than the robot can be detected at least partially when they are visible.

In order to use these characteristics, we divide objects into two categories: those that are smaller than the robot and those that are taller. Locations of the smaller objects can be represented by simply dropping height information of the 3-D segments of the objects because the robot can see all the boundaries of visible surfaces. Therefore, the locations of the smaller objects are acquired by squeezing the 3-D line segments of the smaller objects to the floor. For the taller objects, the robot may not see all the boundaries of visible surfaces. Horizontal boundary lines generated by an object and either the floor or the ceiling may not be seen due to occlusion by other objects or the small viewing angle of a lens. However, the vertical boundary lines are likely to be seen in most cases because they are tall enough not to be occluded when they are within viewing angles. These vertical lines are important clues to find the loca-

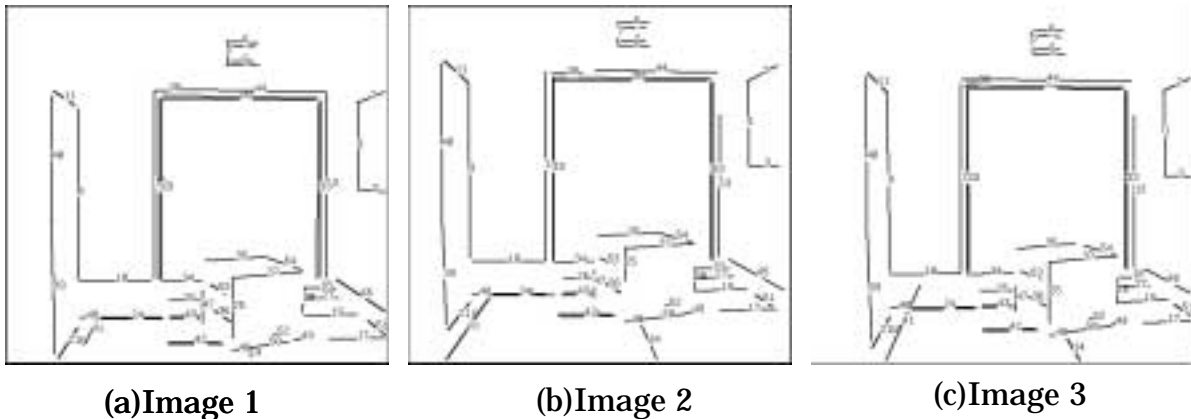


Figure 9.2 Matched segments for Laboratory 1

image is segmented into vertical slices according to the vertical segments. Then segments are distributed among the vertical slices. Now each vertical slice is verified if it can make a vertical surface. The verification is simply done using the viewing triangle constraint and the stability constraint, described in Section 9.2. While the validity of a vertical surface is decided, the segments in its slice are squeezed into an s-map if they can block robot movement..

These simple verification and mapping algorithms allow the robot to build the s-map efficiently. In addition, the s-map can represent the environment more realistically and completely with relatively few 3-D segments because it makes the utmost use of characteristics of indoor environments. Furthermore, the s-map converts many navigation problems in 3-D to 2-D ones because it can represent objects in 2-D domain. This s-map is used for the robot to find a path and navigate the environments.

In Section 9.2 , we introduce an s-map as a spatial Representation For an indoor robot. In Section 9.3, we analyze the algorithm of the s-map. In Section 9.4 , we present the s-map results for indoor environments. Section 9.5 contains the conclusion of this paper.

9.2 S-map

Robot Navigation is defined by the set of tasks that a mobile robot has to perform in order to reach a goal. The tasks involve sensing the environment, analyzing and representing the environment, path planning, and robot locomotion. In this paper, we focus on the environmental representation for the indoor robot using a passive stereo system when only a *generic* map is given. The passive stereo system provides only 3-D segments for the robot to make a navigable map.

The navigable map should represent where obstacles are located, and may represent the shape of the obstacles, when necessary. However, most indoor robots need just the locations of obstacles because they can move only in a plane. Thus, we propose a spatial representation, s-map, which can represent the locations of obstacles in a



(a) Image 1

(b) Image 2

(c) Image 3

Figure 9.1 Laboratory 1 scene.

We use a Denning mobile robot, Antigone, for our experiments. This robot has ultra sonic sensors that are not used in these experiments, and vision sensors composed of three Sony cameras with Cosmincar lenses of focal length 8mm. The three camera are used to take images. The images are then digitized by a Sun Videopix on a Sun workstation. Then, the edges of the images are detected by a Canny edge detector [152].

We use a junction-supported trinocular stereo system for matching the segments. Our method similar to one developed at INRIA [147,148] but incorporates important junction information [155]. Supporting junctions have been introduced for robust matching, and are defined as the junctions of a segment in one image, which are also maintained in homologous segments in the other images. The supporting junction for a segment is formed by the segment and one representative branch of the segment. The representative branch is selected among many branches to reduce complexity. This junction-supported trinocular matching reduces the ambiguity caused by edge detection error, calibration error, and lens distortion [155]. This stereo system gives segment matches as shown in Figure 9.13 that is the results of stereo matching for Figure 9.9 . Now the matched segments are used to reconstruct 3-D segments for making the s-map.

Making a complete map from a sparse depth map, such as Figure 9.1 , is one of the most difficult problems because of insufficient 3-D information to resolve ambiguities in surface reconstruction. As we see in Figure 9.13 , the nearest segments in either 2-D or 3-D do not form a coplanar surface so that Delaunay triangulation methods make unrealistic surfaces. Moreover, deficient 3-D segments make surfaces shrunk. This problem is severer in conceptual grouping methods. The walls in Figure 9.1 do not have enough segments, so that they can hardly recovered. In addition, the perceptual grouping is not so fast that can be used in robot navigation.

From reconstructed 3-D segments and an image, the s-map represents the locations of obstacles in a planar domain by simple mapping and verification. At first, an

term “Generic map” to refer such knowledge. For a robot with only a *generic* map, the representation should allow the robot to convert a *generic* map to a specific navigable map.

Furthermore, the sensors used influence the representation of the environment. The main sensors used presently can be divided into two categories: active range sensors and passive range sensors. Using active sensors, an elevation map can be easily acquired because the sensors get dense occupancy information. However, they are expensive and the occupancy information is not enough for robot navigation with only a generic map because object recognition is an important task for such a robot to reach a goal that is described symbolically, for example “turn at the end of a corridor.” In the second category, passive stereo can acquire 3-D information of the environment. With the passive stereo, we get only a very sparse depth map. Making a complete map from the sparse depth map is one of the most challenging problems in computer vision.

Little research has been done on the environmental representation for indoor robot navigation with a *generic* map using passive sensors. The current methods for such environmental representation fall into two categories: those methods without surface reconstruction and those methods with surface reconstruction. In the first category, Moravec represented an environment simply with detected features [159], and Brauneegg proposed a grid-based presentation using only vertical features [150]. Further, [156,160] tried to represent an environment with a single horizontal slice of a whole 3-D environment. These methods, which do not have surface reconstruction, make incomplete maps. This is because they may miss many surfaces if the surfaces do not have sufficient features or vertical lines. In addition, the method using a single horizontal slice suffers when all horizontal slices of the 3-D space are not the same. For the second category, Bras-Mehlman et al. attempted to make a surface with 3-D Delaunay triangulation and a visibility condition [149,157]. Bruzzone *et al.* first made 2-D Delaunay triangulation then back-projected into 3-D space [151]. Because these methods try to make a surface from adjacent segments, the reconstructed surfaces may be unreasonable, and the constructed free space is shrunk unless there are sufficient segments. Conceptual grouping has also been used to reconstruct surfaces. Mohan and Nevatia attempted to make surfaces by finding rectangles [158]. Chung and Nevatia made surfaces from junctions[154]. Although the reconstructed surfaces are more reasonable, these methods need a lot of computation.

We propose a spatial representation, s-map, for indoor robot navigation using a stereo system when a *generic* map is provided. The environments in which we have experimented are laboratories and corridors in our building, and mostly consist of vertical and horizontal surfaces. A laboratory used in the experiments is shown in Figure 9.1 . Obstacles in the environments have regular shapes that can be linearly approximated. The obstacles used in the experiment are traffic cones, trash cans, boxes, cabinets, desks, etc..

9 Representation and Computation of the Spatial Environment for Indoor Navigation

Dongsung Kim and Ramakant Nevatia

We introduce a spatial representation, s-map, for an indoor navigation robot. The s-map represents the locations of obstacles in a planar domain, where obstacles are defined as any objects that can block movement of the robot. In building the s-map, the viewing triangle constraint and the stability constraint are introduced for efficient verification of vertical surfaces. These verified vertical surfaces and 3-D segments of obstacles smaller than a robot, are mapped to the s-map by simply dropping height information. Thus, the s-map is made directly from 3-D segments with simple verification, and represents obstacles in a planar domain so that it becomes a navigable map for the robot without further processing. In addition to efficient map building, the s-map represents the environment more realistically and completely. Furthermore, the s-map converts many navigation problems in 3-D, such as map fusion and path planning, into 2-D ones. We present the analysis of the s-map in terms of complexity and reliability, and discuss its pros and cons. Moreover, we show the results of the s-maps for indoor environments.

9.1 Introduction

Most navigation robots use vision systems to acquire information about the environment in which they navigate. To acquire information, a robot should be able to represent the environment in some way. This depends on various conditions: the tasks to perform, the amount of an *a priori* knowledge of the environment, and the sensors used. A complete and accurate representation is needed as a robot performs sophisticated tasks. For instance, an occupancy map is enough to avoid obstacles, while explicit representation of materials is necessary for landmark recognition.

The amount of *a priori* knowledge for the environment determines the completeness of representation of the environment. A fairly complete representation is necessary as a robot has small *a priori* knowledge for the environment. For a robot navigating in a known environment, the environmental representation can be simple low level features, such as edges in 2-D images, because the necessary condition for the robot to reach a goal is to locate itself in the environment. However, when a robot is navigating without a precise map, the environmental representation needs to be more complex. In our case, the robot has general information about the environment, such as flat floors and vertical walls. However, it does not have the specifics of the particular environment; the locations of walls or doors and their widths. We will use the