Finding Clusters and Planes from 3D Line Segments with Application to 3D Motion Determination*

Zhengyou Zhang and Olivier D. Faugeras

INRIA Sophia-Antipolis, 2004 route des Lucioles, F-06565 Valbonne Cedex, France

Abstract. We address in this paper how to find clusters based on proximity and planar facets based on coplanarity from 3D line segments obtained from stereo. The proposed methods are efficient and have been tested with many real stereo data. These procedures are indispensable in many applications including scene interpretation, object modeling and object recognition. We show their application to 3D motion determination. We have developed an algorithm based on the hypothesize-and-verify paradigm to register two consecutive 3D frames and estimate their transformation/motion. By grouping 3D line segments in each frame into clusters and planes, we can reduce effectively the complexity of the hypothesis generation phase.

1 Introduction

We address in this paper how to divide 3D line segments obtained from stereo into geometrically compact entities (clusters) and coplanar facets. Such procedures are indispensable in many applications including scene interpretation, object modeling and object recognition. We show in this paper their application to 3D motion determination. The problem arises in the context of a mobile robot navigating in an unknown indoor scene where an arbitrary number of rigid mobile objects may be present. Although the structure of the environment is unknown, it is assumed to be composed of straight line segments. Many planar faced objects are expected in an indoor scene. A stereo rig is mounted on the mobile robot. Our current stereo algorithm uses three cameras and has been described in [1,2]. The 3D primitives used are line segments corresponding to significant intensity discontinuities in the images obtained by the stereo rig.

Given two sets of primitives observed in two views, the task of matching is to establish a correspondence between them. By a correspondence, we mean that the two paired primitives are the different observations (instances) of a single primitive undergoing motion. The matching problem has been recognized as a very difficult problem. It has an exponential complexity in general. The rigidity assumption about the environment and objects is used in most matching algorithms. We have developed an algorithm using the rigidity constraints to guide a hypothesize-and-verify method. Our idea is simple. We use the rigidity constraints to generate some hypothetical primitive correspondences between two successive frames. The rigidity constraints include the invariants of the length of a segment, the distance and angle between two segments, etc. Due to inherent noise in measurements, the equalities in the rigidity constraints are not true anymore. Unlike other methods using some empirical fixed thresholds, we determine dynamically the threshold from the error measurements of 3D data given by a stereo system. Two pairings of 3D line segments form a matching hypothesis if they satisfy the rigidity constraints. Since we apply the rigidity constraints to any pair of segments, if we explore all possible pairs, the number of operations required is $\binom{m}{2}\binom{n}{2}2!$, where m is the number of segments in the first frame and n that in the second frame. Therefore, the complexity is $O(m^2n^2)$.

* This work was supported in part by Esprit project P940.

For each hypothesis, we compute an initial estimate of the displacement using the iterative extended Kalman filter. We then evaluate the validity of these hypothetical displacements. We apply this estimate to the first frame and compare the transformed frame with the second frame. If a transformed segment of the first frame is similar enough to a segment in the second frame, this pairing is considered as matched and the extended Kalman filter is again used to update the displacement estimation. After all segments have been processed, we obtain, for each hypothesis, the optimal estimate of motion, the estimated error given by the filter and the number of matches. As one can observe, the complexity of verifying each hypothesis is $O(mn)$. Once all hypotheses are evaluated, the hypothesis which gives the minimal estimated error and the largest number of matches is considered as the best one, and its corresponding optimal estimate is retained as the displacement between the two frames. Due to the rigidity constraints, the number of hypothetical displacements is usually very small, and computational efficiency is achieved. We exploit the rigidity constraint locally in the hypothesis generation phase and globally in the hypothesis verification phase, as shown later. Figure 1 illustrates diagrammatically the principle of our hypothesize-and-verify method.

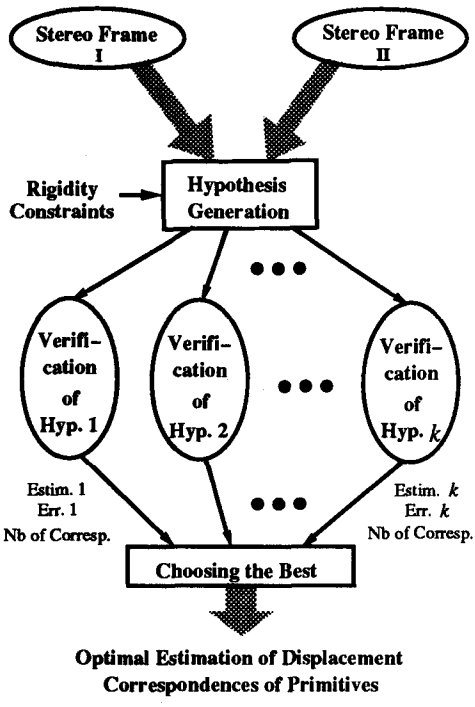

Fig. 1. Diagram of the displacement analysis algorithm based on the hypothesize-and-verify paradigm

A number of methods have been used in our implementation of the above method to reduce the complexities of both the hypothesis generation and verification stages. The interested reader is referred to [3,4]. In this paper, we show how grouping can reduce the complexity of the hypothesis generation stage. We use proximity and coplanarity to organize segments into geometrically compact clusters and planes. These procedures are also useful to scene interpretation.

2 Grouping Speeds Up the Hypothesis Generation Process

If we can divide segments in each frame into several groups such that segments in each group are likely to have originated from a single object, we can considerably speed up the hypothesis generation process. This can be seen from the following. Assume we have divided the two consecutive frames into g_1 and g_2 groups. For simplicity, assume each group in a frame contains the same number of segments. Thus a group in the first frame, G_{1i}, has m/g_1 segments, and a group in the second frame, G_{2i}, has n/g_2 segments, where m and n are the total numbers of segments in the first and second frames. Possible matches of the segments in G_{1i} $(i = 1, \ldots, g_1)$ are restricted in one of the G_{2j}'s $(j = 1, \ldots, g_2)$. That is, it is not possible that one segment in G_{1i} is matched to one segment in G_{2j} and

another segment in the same G_{1i} is matched to one segment in G_{2k} ($k \neq j$). We call this condition the *completeness of grouping*. We show in Fig. 2 a noncomplete grouping in the second frame. The completeness of a grouping implies that we need only apply the hypothesis generation process to each pairing of groups. As we have $g_1 \times g_2$ such pairings and that the complexity for each paring is $O((\frac{m}{g_1})^2(\frac{n}{g_2})^2)$, the total complexity is now $O(\frac{m^2n^2}{g_1g_2})$. Thus we have a speedup of $O(g_1g_2)$.

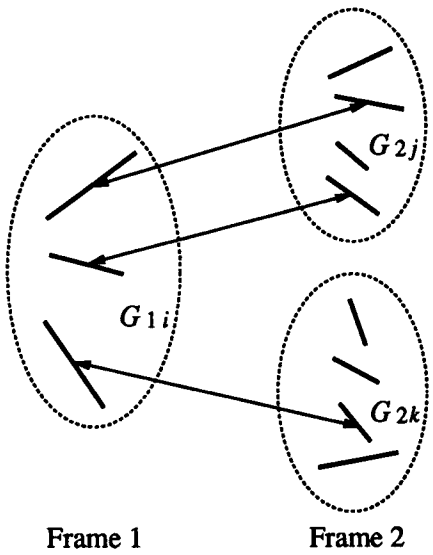

Fig. 2. A grouping which is not complete

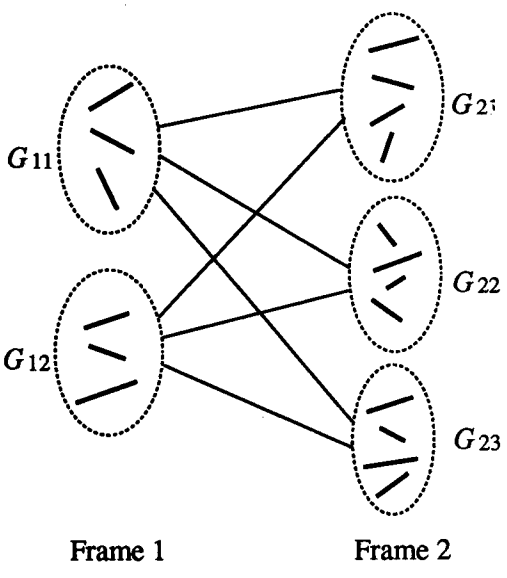

Fig. 3. Illustration of how grouping speeds up the hypothesis generation process

Take a concrete example (see Fig. 3). We have 6 and 12 segments in Frame 1 and Frame 2, respectively. If we directly apply the hypothesis generation algorithm to these two frames, we need $6 \times 5 \times 12 \times 11/2 = 1980$ operations. If the first frame is divided into 2 and the second into 3, we have 6 pairings of groups. Applying the hypothesis generation algorithm to each pairing requires $3 \times 2 \times 4 \times 3/2 = 36$ operations. The total number of operations is $6 \times 36 = 216$, and we have a speedup of 9. We should remember that the $O(g_1g_2)$ speedup is achieved at the cost of a prior grouping process. Whether the speedup is significant depends upon whether the grouping process is efficient.

One of the most influential grouping techniques is called the *perceptual grouping*, pioneered by Low [5]. In our algorithm, grouping is performed based on proximity and coplanarity of 3D line segments. Use of proximity allows us to roughly divide the scene into clusters, each constituting a geometrically compact entity. As we mainly deal with indoor environment, many polyhedral objects can be expected. Use of coplanarity allows us to further divide each cluster into semantically meaningful entities, namely planar facets.

3 Finding Clusters Based on Proximity

Two segments are said to be *proximally connected* if one segment is in the neighborhood of the other one. There are many possible definitions of a neighborhood. Our definition is: the neighborhood of a segment S is a cylindrical space C with radius r whose axis is coinciding with the segment S. This is shown in Fig. 4. The top and bottom of the cylinder

are chosen such that the cylinder C contains completely the segment S. S intersects the two planes at A and B. The segment AB is called the *extended segment* of S. The distance from one endpoint of S to the top or bottom of the cylinder is b. Thus the volume V of the neighborhood of S is equal to $\pi r^2(l + 2b)$, where l is the length of S. We choose $b = r$. The volume of the neighborhood is then determined by r.

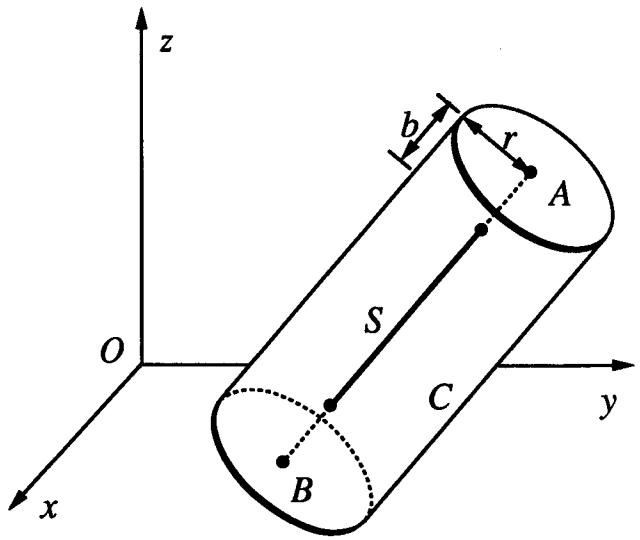

Fig. 4. Definition of a segment's neighborhood

A segment S_i is said in the neighborhood of S if S_i intersects the cylindrical space C. From this definition, S_i intersects C if either of the following condition is satisfied:

1. At least one of the endpoints of S_i is in C.
2. The distance between the supporting lines of S and S_i is less than r and the common perpendicular intersects both S_i and the *extended* segment of S.

We define a *cluster* as a group of segments, every two of which are proximally connected in the sense defined above either directly or through one or more segments in the same group. A simple implementation to find clusters by testing the above conditions leads to a complexity of $O(n^2)$ in the worst case, where n is the number of segments in a frame. In the following, we present a method based on a *bucketing technique* to find clusters. First, the minima and maxima of the x, y and z coordinates are computed, denoted by $x_{\min}$, $y_{\min}$, $z_{\min}$ and $x_{\max}$, $y_{\max}$, $z_{\max}$. Then the parallelepiped formed by the minima and maxima is partitioned into p^3 buckets W_{ijk} ($p = 16$ in our implementation). To each bucket W_{ijk} we attach the list of segments L_{ijk} intersecting it. The key idea of bucketing techniques is that on the average the number of segments intersecting a bucket is much smaller than the total number of segments in the frame. The computation of attaching segments to buckets can be performed very fast by an algorithm whose complexity is linear in the number of segments. Finally, a recursive search is performed to find clusters. We can write an algorithm to find a cluster containing segment S in pseudo C codes as:

```
List find_cluster(S)
Segment S ;
```

```
{
    List cluster = NULL ;
    if is_visited(S) return NULL ;
    mark_segment_visited(S) ;
    list_buckets = find_buckets_intersecting_neighborhood_of(S) ;
    list_segments = union_of_all_segments_in(list_buckets) ;
    for (S_i = each_segment_in(list_segments))
        cluster = cluster ⋃ {S_i} ⋃ find_cluster(S_i) ;
    return cluster ;
}
```

where `List` is a structure storing a list of segments, defined as

```
struct cell {
        Segment seg ;
        struct cell *next ;
} *List ;
```

From the above discussion, we see that the operations required to find a cluster are really very simple with the aide of a bucketing technique, except probably the function **find_buckets_intersecting_neighborhood_of**(S). This function, as indicated by its name, should find all buckets intersecting the neighborhood of the segment S. That is, we must examine whether a bucket intersects the cylindrical space C or not, which is by no means a simple operation. The gain in efficiency through using a bucketing technique may become nonsignificant. Fortunately, we have a very good approximation as described below which allows for an efficient computation.

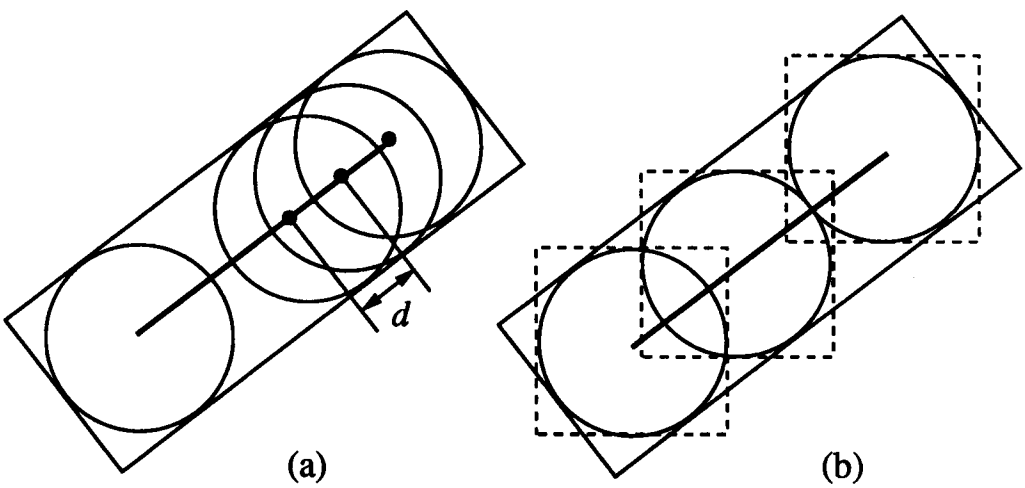

Fig. 5. Approximation of a neighborhood of a segment

Fill the cylindrical space C with many spheres, each just fitting into the cylinder, i.e., whose radius is equal to r. We allow intersection between spheres. Fig. 5a illustrates the situation using only a section passing through the segment S. The union of the set of spheres gives an approximation of C. When the distance d between successive sphere centers approaches to zero (i.e., $d \rightarrow 0$), the approximation is almost perfect, except in the top and bottom of C. The error of the approximation in this case is $\frac{1}{3}\pi r^3$. This part is not very important because it is the farthest to the segment. Although the operation

to examine the intersectionness between a bucket and a sphere is simpler than between a bucket and a cylinder, it is not beneficial if we use too many spheres. What we do is further, as illustrated in Fig. 5b. Spheres are not allowed to intersect with each other (i.e., $d = 2r$) with the exception of the last sphere. The center of the last sphere is always at the endpoint of S, so it may intersects with the previous sphere. The number of spheres required is equal to $\lceil \frac{l}{2r} + 1 \rceil$, where $\lceil a \rceil$ denotes the smallest integer greater than or equal to a. It is obvious that the union of these spheres is always smaller than the cylindrical space C. Now we replace each sphere by a cube circumscribing it and aligned with the coordinate axes (represented by a dashed rectangle in Fig. 5b). Each cube has a side length of $2r$. It is now almost trivial to find which buckets intersect a cube. Let the center of the cube be $[x,\ y,\ z]^T$. Let

$$
\begin{aligned}
i_{\min} &= \max[0, \lfloor (x - x_{\min} - r)/dx \rfloor] , & i_{\max} &= \min[m-1, \lceil (x - x_{\min} + r)/dx - 1 \rceil] , \\
j_{\min} &= \max[0, \lfloor (y - y_{\min} - r)/dy \rfloor] , & j_{\max} &= \min[m-1, \lceil (y - y_{\min} + r)/dy - 1 \rceil] , \\
k_{\min} &= \max[0, \lfloor (z - z_{\min} - r)/dz \rfloor] , & k_{\max} &= \min[m-1, \lceil (z - z_{\min} + r)/dz - 1 \rceil] ,
\end{aligned}
$$

where $\lfloor a \rfloor$ denotes the greatest integer less than or equal to a, m is the dimension of the buckets, $dx = (x_{\max} - x_{\min})/m$, $dy = (y_{\max} - y_{\min})/m$, and $dz = (z_{\max} - z_{\min})/m$. The buckets intersecting the cube is simply $\{W_{ijk} \mid i = i_{\min}, \ldots, i_{\max}, j = j_{\min}, \ldots, j_{\max}, k = k_{\min}, \ldots, k_{\max}\}$.

If a segment is parallel to either x, y or z axis, it can be shown that the union of the cubes is bigger than the cylindrical space C. However, if a segment is near 45°(or 135°) to an axis (as shown in Fig. 5b), there are some gaps in approximating C. These gaps are usually filled by buckets, because a whole bucket intersecting one cube is now considered as part of the neighborhood of S.

4 Finding Planes

Several methods have been proposed in the literature to find planes. One common method is to directly use data from range finders [6,7]. The expert system of Thonnat [8] for scene interpretation is capable of finding planes from 3D line segments obtained from stereo. The system being developed by Grossmann [9,10] aims at extracting, also from 3D line segments, visible surfaces including planar, cylindrical, conical and spherical ones. In the latter system, each coplanar crossing pair of 3D line segments (i.e., they are neither collinear nor parallel) forms a candidate plane. Each candidate plane is tested for compatibility with the already existing planes. If compatibility is established, the existing plane is updated by the candidate plane. In this section, we present a new method to find planes from 3D line segments.

As we do not know how many and where are the planes, we first try to find two coplanar line segments which can define a plane, and then try to find more line segments which lie in this hypothetical plane until all segments are processed. The segments in this plane are marked visited. For those unvisited segments, we repeat the above process to find new planes.

Let a segment be represented by its midpoint $\mathbf{m}$, its unit direction vector $\mathbf{u}$ and its length l. Because segments reconstructed from stereo are always corrupted by noise, we attach to each segment an uncertainty measure (covariance matrix) of $\mathbf{m}$ and $\mathbf{u}$, denoted by $\Lambda_{\mathbf{m}}$ and $\Lambda_{\mathbf{u}}$. The uncertainty measure of l is not required. For two noncollinear segments: $(\mathbf{m}_1, \mathbf{u}_1)$ with $(\Lambda_{\mathbf{m}_1}, \Lambda_{\mathbf{u}_1})$ and $(\mathbf{m}_2, \mathbf{u}_2)$ with $(\Lambda_{\mathbf{m}_2}, \Lambda_{\mathbf{u}_2})$, the coplanarity condition is

$$c \triangleq (\mathbf{m}_2 - \mathbf{m}_1) \cdot (\mathbf{u}_1 \wedge \mathbf{u}_2) = 0 , \tag{1}$$

where $\cdot$ and $\wedge$ denote the dot product and the cross product of two vectors, respectively.

In reality, the condition (1) is unlikely met. Instead, we impose that $|c|$ is less than some threshold. We determine the threshold in a dynamic manner by relating it to the uncertainty measures. The variance of c, denoted by Λ_c, is computed from the covariance matrices of the two segments by

$$\Lambda_c = (\mathbf{u}_1 \wedge \mathbf{u}_2)^T(\Lambda_{\mathbf{m}_1} + \Lambda_{\mathbf{m}_2})(\mathbf{u}_1 \wedge \mathbf{u}_2) + [\mathbf{u}_1 \wedge (\mathbf{m}_2 - \mathbf{m}_1)]^T \Lambda_{\mathbf{u}_2}[\mathbf{u}_1 \wedge (\mathbf{m}_2 - \mathbf{m}_1)] + [\mathbf{u}_2 \wedge (\mathbf{m}_2 - \mathbf{m}_1)]^T \Lambda_{\mathbf{u}_1}[\mathbf{u}_2 \wedge (\mathbf{m}_2 - \mathbf{m}_1)] .$$

Here we assume there is no correlation between the two segments. Since c^2/Λ_c follows a χ^2 distribution with one degree of freedom, two segments are said coplanar if

$$c^2/\Lambda_c \leq \kappa , \tag{2}$$

where κ can be chosen by looking up the χ^2 table such that $\Pr(\chi^2 \leq \kappa) = \alpha$. In our implementation, we set $\alpha = 50\%$, or $\kappa = 0.5$.

As discussed in the last paragraph, the two segments used must not be collinear. Two segments are collinear if and only if the following two conditions are satisfied:

$$\mathbf{u}_1 - \mathbf{u}_2 = \mathbf{0} , \quad \text{and} \quad \mathbf{u}_1 \wedge (\mathbf{m}_2 - \mathbf{m}_1) = \mathbf{0} . \tag{3}$$

The first says that two collinear segments should have the same orientation (Remark: segments are oriented in our stereo system). The second says that the midpoint of the second segment lies on the first segment. In reality, of course, these conditions are rarely satisfied. A treatment similar to the coplanarity can be performed.

Once two segments are identified to lie in a single plane, we estimate the parameters of the plane. A plane is described by

$$ux + vy + wz + d = 0 , \tag{4}$$

where $\mathbf{n} = [u, v, w]^T$ is parallel to the normal of the plane, and $|d|/\|\mathbf{n}\|$ is the distance of the origin to the plane. It is clear that for an arbitrary scalar $\lambda \neq 0$, $\lambda[u, v, w, d]^T$ describes the same plane as $[u, v, w, d]^T$. Thus the minimal representation of a plane has only three parameters. One possible minimal representation is to set $w = 1$, which gives

$$ux + vy + z + d = 0 .$$

However, it cannot represent planes parallel to the z-axis. To represent all planes in 3D space, we should use three maps [11]:

Map 1: $ux + vy + z + d = 0$ for planes nonparallel to the z-axis, (5)

Map 2: $x + vy + wz + d = 0$ for planes nonparallel to the x-axis, (6)

Map 3: $ux + y + wz + d = 0$ for planes nonparallel to the y-axis, (7)

In order to choose which map to be used, we first compute an initial estimate of the plane normal $\mathbf{n}_0 = \mathbf{u}_1 \wedge \mathbf{u}_2$. If the two segments are parallel, $\mathbf{n}_0 = \mathbf{u}_1 \wedge (\mathbf{m}_2 - \mathbf{m}_1)$. If the z component of $\mathbf{n}$ has a maximal absolute value, Map 1 (5) will be used; if the x component has a maximal absolute value, Map 2 (6) will be used; otherwise, Map 3 (7) will be used. An initial estimate of d can then be computed using the midpoint of a segment. In the sequel, we use Map 1 for explanation. The derivations are easily extended to the other maps.

We use an extended Kalman filter [12] to estimate the plane parameters. Let the state vector be $\mathbf{x} = [u, v, d]^T$. We have an initial estimate $\mathbf{x}_0$ available, as described just above. Since this estimate is not good, we set the diagonal elements of the initial covariance matrix $\Lambda_{\mathbf{x}_0}$ to a very big number and the off-diagonal elements to zero. Suppose a 3D segment with parameters $(\mathbf{u}, \mathbf{m})$ is identified as lying in the plane. Define the measurement vector as $\mathbf{z} = [\mathbf{u}^T, \mathbf{m}^T]^T$. We have two equations relating $\mathbf{z}$ to $\mathbf{x}$:

$$\mathbf{f}(\mathbf{x}, \mathbf{z}) = \begin{cases} \mathbf{n}^T\mathbf{u} \\ \mathbf{n}^T\mathbf{m} + d \end{cases} = \mathbf{0} , \tag{8}$$

where $\mathbf{n} = [u, v, 1]^T$. The first equation says that the segment is perpendicular to the plane normal, and the second says that the midpoint of the segment is on the plane. In order to apply the Kalman filter, we must linearize the above equation [13], which gives

$$\mathbf{y} = M\mathbf{x} + \boldsymbol{\xi} \ , \tag{9}$$

where $\mathbf{y}$ is the new measurement vector, M is the observation matrix, and $\boldsymbol{\xi}$ is the noise disturbance in $\mathbf{y}$, and they are given by

$$M = \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{z})}{\partial \mathbf{x}} = \begin{bmatrix} u_x & u_y & 0 \\ m_x & m_y & 1 \end{bmatrix} \ , \tag{10}$$

$$\mathbf{y} = -\mathbf{f}(\mathbf{x}, \mathbf{z}) + M\mathbf{x} = \begin{bmatrix} -u_x \\ -m_z \end{bmatrix} \ , \tag{11}$$

$$\boldsymbol{\xi} = \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{z})}{\partial \mathbf{z}} \boldsymbol{\delta}_\mathbf{z} = \begin{bmatrix} \mathbf{n}^T & \mathbf{0}^T \\ \mathbf{0}^T & \mathbf{n}^T \end{bmatrix} \boldsymbol{\delta}_\mathbf{z} \ , \tag{12}$$

where $\mathbf{0}$ is the 3D zero vector and $\boldsymbol{\delta}_\mathbf{z}$ is the noise disturbance in $\mathbf{z}$. Now that we have two segments which have been identified to be coplanar and an initial estimate of the plane parameters, we can apply the extended Kalman filter based on the above formulation to obtain a better estimate of $\mathbf{x}$ and its error covariance matrix $\Lambda_\mathbf{x}$.

Once we have estimated the parameters of the plane, we try to find more evidences of the plane, i.e., more segments in the same plane. If a 3D segment $\mathbf{z} = [\mathbf{u}^T, \mathbf{m}^T]^T$ with $(\Lambda_\mathbf{u}, \Lambda_\mathbf{m})$ lies in the plane, it must satisfy Eq. (8). Since data are noisy, we do not expect to find a segment having exactly $\mathbf{p} \triangleq \mathbf{f}(\mathbf{x}, \mathbf{z}) = 0$. Instead, we compute the covariance matrix $\Lambda_\mathbf{p}$ of $\mathbf{p}$ as follows:

$$\Lambda_\mathbf{p} = \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{z})}{\partial \mathbf{x}} \Lambda_\mathbf{x} \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{z})}{\partial \mathbf{x}}^T + \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{z})}{\partial \mathbf{z}} \Lambda_\mathbf{z} \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{z})}{\partial \mathbf{z}}^T \ , \tag{13}$$

where $\frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{z})}{\partial \mathbf{x}}$ and $\frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{z})}{\partial \mathbf{z}}$ are computed by Eqs. (10) and (12), respectively. If

$$\mathbf{p}^T \Lambda_\mathbf{p}^{-1} \mathbf{p} \leq \kappa_p \ , \tag{14}$$

then the segment is considered as lying in the plane. Since $\mathbf{p}^T \Lambda_\mathbf{p}^{-1} \mathbf{p}$ follows a χ^2 distribution with 2 degrees of freedom, we can choose an appropriate κ_p by looking up the χ^2 table such that $\Pr(\chi^2 \leq \kappa_p) = \alpha_p$. We choose $\alpha_p = 50\%$, or $\kappa_p = 1.4$. Each time we find a new segment in the plane, we update the plane parameters $\mathbf{x}$ and $\Lambda_\mathbf{x}$ and try to find still more. Finally, we obtain a set of segments supporting the plane and also an estimate of the plane parameters accounting for all these segments.

5 Experimental Results

In this section we show the results of grouping using an indoor scene. A stereo rig takes three images, one of which is displayed in Fig. 6. After performing edge detection, edge linking and linear segment approximation, the three images are supplied to a trinocular stereo system, which reconstructs a 3D frame consisting of 137 3D line segments. Figure 7 shows the front view (projection on the plane in front of the stereo system and perpendicular to the ground plane) and the top view (projection on the ground plane) of the reconstructed 3D frame.

We then apply the bucketing technique to this 3D frame to sort segments into buckets, which takes about 0.02 seconds of user time on a Sun 4/60 workstation. The algorithm described in Sect. 3 is then applied, which takes again 0.02 seconds of user time to find two clusters. They are respectively shown in Figs. 8 and 9. Comparing these with Fig. 7, we observe that the two clusters do correspond to two geometrically distinct entities.

Fig. 6. Image taken by the first camera

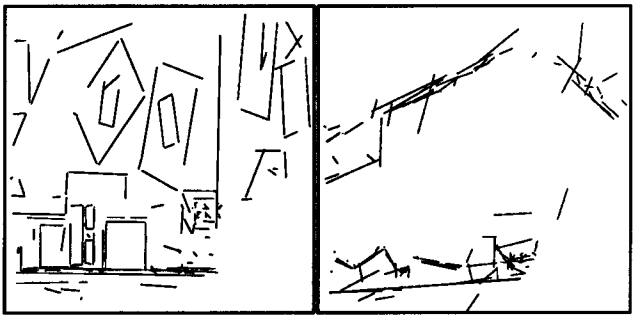

Fig. 7. Front and top views of the reconstructed 3D frame

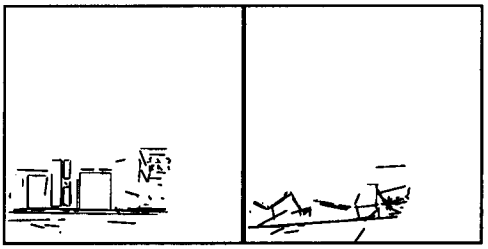

Fig. 8. Front and top views of the first cluster

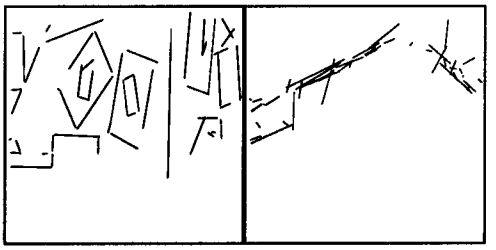

Fig. 9. Front and top views of the second cluster

Finally we apply the algorithm described in Sect. 4 to each cluster, and it takes 0.35 seconds of user time to find in total 11 planes. The four largest planes contain 17, 10, 25 and 13 segments, respectively, and they are shown in Figs. 10 to 13. Other planes contain less than 7 segments, corresponding to the box faces, the table and the terminal. From these results, we observe that our algorithm can reliably detect planes from 3D line segments obtained from stereo, but a plane detected does not necessarily correspond to a physical plane. The planes shown in Figs. 11 to 13 correspond respectively to segments on the table, the wall and the door. The plane shown in Fig. 10, however, is composed of segments from different objects, although they do satisfy the coplanarity. This is because in our current implementation any segment in a *cluster* satisfying the coplanarity is retained as a support of the plane. One possible solution to this problem is to grow the plane by look for segments in the *neighborhood* of the segments already retained as supports of the plane.

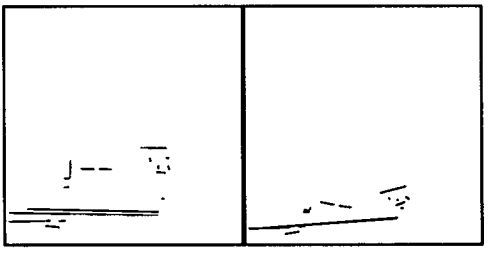

Fig. 10. Front and top views of the first plane

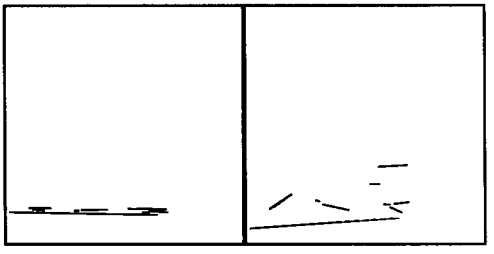

Fig. 11. Front and top views of the second plane

Due to space limitation, the reader is referred to [4] and [13] for application to 3D motion determination.

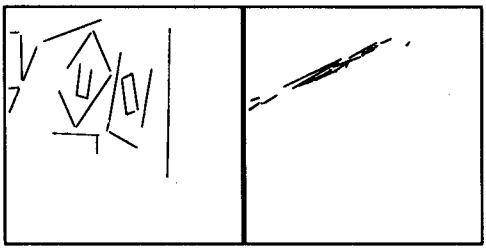

Fig. 12. Front and top views of the third plane

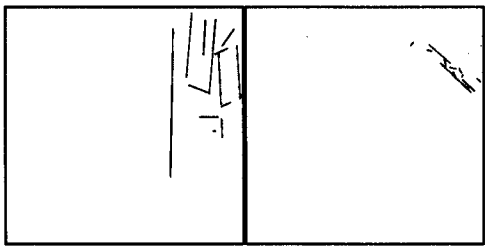

Fig. 13. Front and top views of the fourth plane

6 Conclusion

We have described how to speed up the motion determination algorithm through grouping. A formal analysis has been done. A speedup of $O(g_1 g_2)$ can be achieved if two consecutive frames have been segmented into g_1 and g_2 groups. Grouping must be *complete* in order not to miss a hypothesis in the hypothesis generation process. Two criteria satisfying the completeness condition have been proposed, namely proximity to find clusters which are geometrically compact and coplanarity to find planes. Implementation details have been described. Many real stereo data have been used to test the algorithm and good results have been obtained. We should note that the two procedures are also useful to scene interpretation.

References

1. F. Lustman, *Vision Stéréoscopique et Perception du Mouvement en Vision Artificielle.* PhD thesis, University of Paris XI, Orsay, Paris, France, December 1987.
2. N. Ayache, *Artificial Vision for Mobile Robots: Stereo Vision and Multisensory Perception.* MIT Press, Cambridge, MA, 1991.
3. Z. Zhang, O. Faugeras, and N. Ayache, "Analysis of a sequence of stereo scenes containing multiple moving objects using rigidity constraints," in *Proc. Second Int'l Conf. Comput. Vision*, (Tampa, FL), pp. 177–186, IEEE, December 1988.
4. Z. Zhang and O. D. Faugeras, "Estimation of displacements from two 3D frames obtained from stereo," Research Report 1440, INRIA Sophia-Antipolis, 2004 route des Lucioles, F-06565 Valbonne cedex, France, June 1991.
5. D. Lowe, *Perceptual Organization and Visual Recognition.* Kluwer Academic, Boston, MA, 1985.
6. W. Grimson and T. Lozano-Perez, "Model-based recognition and localization from sparse range or tactile data," *Int'l J. Robotics Res.*, vol. 5, pp. 3–34, Fall 1984.
7. O. Faugeras and M. Hebert, "The representation, recognition, and locating of 3D shapes from range data," *Int'l J. Robotics Res.*, vol. 5, no. 3, pp. 27–52, 1986.
8. M. Thonnat, "Semantic interpretation of 3-D stereo data: Finding the main structures," *Int'l J. Pattern Recog. Artif. Intell.*, vol. 2, no. 3, pp. 509–525, 1988.
9. P. Grossmann, "Building planar surfaces from raw data," Technical Report R4.1.2, ESPRIT Project P940, 1987.
10. P. Grossmann, "From 3D line segments to objects and spaces," in *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, (San Diego, CA), pp. 216–221, 1989.
11. O. D. Faugeras, *Three-Dimensional Computer Vision.* MIT Press, Cambridge, MA, 1991. to appear.
12. P. Maybeck, *Stochastic Models, Estimation and Control.* Vol. 2, Academic, New York, 1982.
13. Z. Zhang, *Motion Analysis from a Sequence of Stereo Frames and its Applications.* PhD thesis, University of Paris XI, Orsay, Paris, France, 1990. in English.

This article was processed using the LaTeX macro package with ECCV92 style