Semi-metric Formal 3D Reconstruction from Perspective Sketches

Alex Sosnov[1], Pierre Macé[2], and Gérard Hégron[3]

[1] Ecole des Mines de Nantes, Département d'Informatique,
44307 Nantes Cedex 3, France
[2] Tornado Technologies Inc., 7–11 av. Raymond Feraud, 06200 Nice, France
[3] CERMA UMR CNRS 1563, Ecole d'Architecture de Nantes,
44319 Nantes Cedex 3, France

Abstract. We present a new approach for accurate and fast reconstruction of 3D models from hand-drawn *perspective* sketches and imposed geometric constraints. A distinctive feature of the approach is the decomposition of the reconstruction process into the stages of correction of the 2D sketch and elevation of the 3D model. All 3D *constraints* that describe the spatial structure of the model are *strictly* satisfied, while *preferences* that describe the model projection are treated in *relaxed* manner. The *constraints* are subdivided into the projective, affine and metric ones and expressed in algebraic form by using the Grassmann-Cayley algebra. The constraints are resolved one by another following the order of their types by using the *local* propagation methods. The *preferences* allow to apply linear approximations and to systematically use formal methods.

1 Introduction

Problems of using traditional conception tools in conjunction with new computer-aided design, modeling and engineering instruments arise nowadays in many domains close to architectural design. Thus, the task of analyzing and understanding 3D shapes from freehand drawings is receiving increased attention.

During the last years, several methods for reconstruction of 3D scenes either from hand-drawn sketches or photographs have been proposed. To reconstruct a 3D scene from a sketch, one seeks, by using numerical methods, the position of the center of projection and the 3D model so that its projection is maximally fits to the sketch ([3], [8]). Such approaches include the analysis of the sketch and even intentions of the user [3] to determine correct forms that the user means to do and spatial constraints that the reconstructing 3D model have to satisfy. However, most of these methods involve axonometric projections, while perspective ones provide more 3D information. On the other hand, the methods [2], [5] allow to reconstruct 3D models from perspective photographs. Moreover, the system [2] provide the user with a set of 3D primitives and allows to set constraints among them. However, such approaches can not be applied to imprecise hand-drawn sketches. All the above methods rely on heavy numerical computations that are time consuming and subject to numerical instabilities.

P.M.A. Sloot et al. (Eds.): ICCS 2002, LNCS 2330, pp. 285−294, 2002.

In this paper, we demonstrate that the problem of reconstruction of 3D models from perspective sketches can be significantly simplified by serializing the resolution of constraints and using geometric algebra. Our approach allows to perform the reconstruction faster and more accurately than before even for imprecise drawings and provides new possibilities for organization of user interfaces.

2 The Approach

Our method of reconstruction of 3D models from perspective hand-drawn sketches is based on separation of **constraints** and **preferences** [4], **serialization** of constraints and their **local** resolution, and application of **formal** methods.

In contrast to other approaches, we do not attempt to find the 3D model that maximally fits to the given sketch. Instead, we demand the user to set constraints that describe spatial structure of the given scene. We find the *correct sketch* that is *merely* close to the original one while *strictly* satisfies all the projective consequences of the imposed constraints. Then we elevate the 3D model from the obtained true projection by using the projective geometric Grassmann-Cayley algebra (Sect. 3). On the other hand, during such an elevation, any 3D element is determined by using its projection only if there is no way to determine it from incident 3D elements. Thus, we rigorously respect all the 3D *constraints* while consider the sketch only as the source of *preferences*.

Each of the 3D constraints represents a projective (*incidence*, *collinearity*), affine (*parallelism* of lines and planes) or metric (*orthogonality* of lines) relation. The constraints are resolved following the order of their types. Namely, we estimate vanishing points and lines to satisfy projective consequences of all the parallelism constraints with the minimum of distortion of the sketch. Then we correct vanishing points and evaluate the position of the center of projection so that all the 3D orthogonality constraints would be satisfied during the elevation of the 3D model. Next, the sketch is redrawn from the corrected vanishing points and certain free points to respect all the projective constraints. Finally, the 3D model is elevated from the obtained true projection and 3D coordinates of certain scene points. This stepwise resolution process is called *serialization*. x Unlike to other approaches, we do not try to solve arising systems of constraints globally. Instead, constraints are presented as a graph and resolved by the *local methods*, mainly by propagation. On the other hand, we use some global information to control propagation processes. Namely, we establish *operation priorities* to choose the most stable and computationally effective solution. Moreover, degenerate local solutions are rejected by using global relation tables. We decompose resolution processes into the *formal* and *numerical* stages and postpone calculations as late as possible. Namely, for most of the serialized stages we firstly construct a formal solution by using only constraint graphs. Next, we evaluate the concrete solution from the obtained formal one and the corresponding numerical parameters. Having the formal solution, we can distinguish between inaccuracies and inconsistencies and rapidly re-evaluate the reconstruction once the user changes numerical parameters or enriches the scene with new details.

3 The Grassman-Cayley Algebra

To build a formal solution of a reconstruction problem in stepwise manner, we need a constructive "coordinate-free" formalism that allows to express projective geometric statements by invariant algebraic ones. Consider a projective space P furnished with a multilinear alternating form called a *bracket*. Using this form, it is possible to define a double algebra of affine subspaces of P (i.e., points, lines and planes) [6]. This algebra is called the *Grassmann-Cayley algebra* (GC algebra). It provides binary operators *join* $\vee$ and *meet* $\wedge$ and the unary *duality* operator *. A *join* represents a sum of disjoint subspaces, while a *meet* represents intersection of subspaces and is dual to the join of the their duals. For example,

$$\begin{gathered}\text{point } a \vee \text{point } b = \text{line } ab \\ (\text{plane } \pi_1 \wedge \text{plane } \pi_2)^* = (\text{plane } \pi_1)^* \vee (\text{plane } \pi_2)^* \\ (a \vee b \vee c \vee d)^* = [a\ b\ c\ d] = s\ .\end{gathered}$$

Since the scalar s identifies the volume, it is impossible to determine a distance in P in terms of this algebra. On the other hand, under certain assumptions it is possible to solve orthogonality problems using the notion of duality. Once we have chosen any plane π to represent the infinite plane π_∞, it is possible to formally represent all the constructions that imply parallelism and certain orthogonality constraints. Therefore our approach is called *semi-metric*.

To perform computations from the GC algebra formal expressions, 3D points are represented by 4D vectors of homogenous coordinates, while 3D lines and planes are represented by antisymmetric matrices containing their Plücker coordinates. All the calculations are reduced to evaluations of exterior products for joins and simple matrix products for meets. These products represent coordinate expressions for the corresponding GC algebra operations. If a result of join or meet is null (i.e., a null tensor), the arguments of the operator are linearly dependent. This fact used is to detect contradictions and imprecisions (Sect. 9).

4 Creation of Geometry

4.1 Elementary Objects and Constraints

We represent any 3D scene with the following elementary objects: *points*, *lines* and *planes*, and constraints: *collinearity* and *coplanarity* of points and lines, *parallelism* and *orthogonality* of lines and planes. The user creates such objects and constraints via high-level primitives or directly on a sketch.

Since we use the Grassman-Cayley algebra, we express any n-ary projective or affine constraint as a set of *incidences*. Incidence (signed below by the symbol ε) is the binary projective relation defined as:

$$\forall A, B \quad A\,\varepsilon\,B \Leftrightarrow A \in B \text{ or } B \in A \text{ or } A \subset B \text{ or } B \subset A\ .$$

Any coplanarity or collinearity constraint just should be binarized, e.g.:

$$\text{points } a, b, c \text{ are collinear } \textbf{iff } \exists \text{ line } L\ \{a\,\varepsilon\,L, b\,\varepsilon\,L, c\,\varepsilon\,L\}\ .$$

Any parallelism can be represented as a triple of incidences:

$$\text{lines } L_1 \parallel L_2 \textbf{ iff } \exists \text{ point } i_\infty \{L_1 \varepsilon i_\infty, L_2 \varepsilon i_\infty, i_\infty \varepsilon \pi_\infty\} \ ,$$

where π_∞ is the infinite plane, while the infinite point i_∞ unambiguously determines the whole pencil of the lines. Therefore, we can present all the projective and affine constraints in the form of a *constraint graph* $CG(V, E)$. Its vertices represent elementary 3D objects while its edges represent established constraints:

$$\forall u, v \in V(CG) \quad (u, v) \in E(CG) \Leftrightarrow u \varepsilon v \ .$$

On the other hand, we do not represent metric constraints by incidences. Instead, we create an *orthogonality graph* $OG(V, E)$ so that its vertices correspond to pencils of parallel lines while its edges represent orthogonalities.

4.2 Consistency of Constraints

We define a *geometry law* as a function from a constraint to its *consequences*, $G : u \varepsilon v \Rightarrow \{x \varepsilon y\}$. For geometry of incidences, we have only 3 basic laws:

1. $L \varepsilon \pi \Rightarrow \forall a \varepsilon L,\ a \varepsilon \pi$
2. $a \varepsilon L \Rightarrow (a \varepsilon \pi,\ b \varepsilon L,\ b \varepsilon \pi \Rightarrow L \varepsilon \pi)$ **and** $\forall \pi \varepsilon L,\ a \varepsilon \pi$
3. $a \varepsilon \pi \Rightarrow (a \varepsilon L,\ b \varepsilon L,\ b \varepsilon \pi \Rightarrow L \varepsilon \pi)$ **and** $(a \varepsilon \tilde{\pi}, L \varepsilon \pi, L \varepsilon \tilde{\pi} \Rightarrow a \varepsilon L)$

where a, L and π are any point, line and plane.

A constraint graph is *saturated* if it contains all the consequences of each of the imposed constraints. Use of only saturated graphs allows to improve efficiency of the reconstruction and to analytically reject degenerate solutions. Moreover, during creation of such a graph it is possible to detect logical contradictions. Thus, a constraint graph is *consistent* if it is saturated and not contradictory.

The consistency is maintained by update procedures that represent the basic geometry laws and are called by the system every time it creates the corresponding incidences. An update procedure infers new constraints that are consequences of the created one and the applied law. Since the geometry laws recursively depend on each other, the update procedures are called recursively. To prevent infinite cycles, one member of the created incidence is blocked as source, other as target so that it is possible to establish new constraints on the target member but not on the source one. Performance of update procedures depends on efficiency of searching geometry law elements and testing whether constraints are already established. Since it is estimated as $O(E/V)$ and for saturated graphs $|E| \gg |V|$, we use global *relation maps* to find constraints in constant time.

4.3 Projection of Constraints

Since projections of 3D points are required for the elevation, they are created directly in the constraint graph. The graph contains the center of projection o and the sketch plane π_{sk}. For any 3D point, its projection $\text{proj}(p)$ and the *line of sight* op are constructed so that $op \varepsilon p$, $op \varepsilon \text{proj}(p)$, $op \varepsilon o$, and $\text{proj}(p) \varepsilon \pi_{\text{sk}}$.

Projections of 3D lines do not provide any information for the reconstruction, while they are required for the resolution of orthogonalities and the sketch correction. Thus, they are created just in the *sketch graph* $SG(V, E)$ that represents the projection of the scene $CG(V, E)$ on the sketch plane:

$$V(SG) = \{\text{proj}(v) : v \in V(CG)\} \textbf{ and } E(SG) = \{(\text{proj}(u), \text{proj}(v)) : u \,\varepsilon\, v\} \ .$$

All the projections and their incidences are constructed by the system every time it creates an usual or infinite 3D point or line, or sets their incidence by following a declared or inferred constraint. Thus, projections of any parallel lines have a common vanishing point. This point is aligned with the center of projection and the infinite point that determines the pencil. Vanishing points of projections of coplanar pencils of parallel lines are aligned.

5 Formal Reconstruction

Once we have built a scene constraint graph, we can construct a formal solution of the reconstruction problem for any scene described by the graph. The solution is represented as the *formal reconstruction plan*. For each of the elementary 3D objects of the scene, the plan contains a coordinate-free expression of the GC algebra determining the object so that it would satisfy the constraints imposed on it. For instance, if a point a has constraints $a \,\varepsilon\, L_1$, $a \,\varepsilon\, L_2$, the plan contains expression $a = L_1 \wedge L_2$, since a meet of two lines defines their intersection.

The algorithm to build a reconstruction plan is based on *propagation of known data*. Indeed, using the GC algebra we can *determine* any 3D object once we have *known* a sufficient number of 3D objects incidents to it. If an object has a redundant number of known neighbors, it can be determined in different ways. To choose the best solution, we set the *operation priorities*. Since the sketch is used just as the source of preferences, any 3D point is determined by using its projection only if it is impossible to determine it via other incident 3D elements. Furthermore, operations of GC algebra differ in accuracy and numerical computation cost. For example, a meet of a line and a plane requires 12 multiplications and is always valid while a meet of two lines requires 36 multiplications and may become degenerate due to precision errors. We obtain the following operations table, where priorities are decrementing from top to bottom, left to right:

line	$a \vee b$	$\pi_1 \wedge \pi_2$			
plane	$L \vee a$	$a \vee b \vee c$			
point	$L \wedge \pi$	$\pi_1 \wedge \pi_2 \wedge \pi_3$	$L_1 \wedge L_2$	$L \wedge op$	$\pi \wedge op$.

To start the algorithm, we should have some known points. The center of projection, all the sketch points (i.e., visible projections of 3D points), and all the vanishing points are set as known and propagated to determine the lines of sight. We also have to know certain 3D objects. The number of these objects is the number of degrees of freedom of the scene. This number and required objects are determined by the algorithm. All the current determined objects are stored

in the priority queue according to priorities of their determining operations. If the queue is empty, the scene is underconstrained. Therefore, we increment the number of degrees of freedom and demand the user for new constraints (on the minimally undetermined objects). An object becomes known once it is extracted from the queue. Its determining operation is added to the reconstruction plan. All the already known objects incidents to the current known one set the alternatives to determine it. These *alternative evaluators* are added to the *alternatives graph* that is used later on the numerical evaluation stage. The current known object is propagated to its unknown (but possibly determined!) neighbors. Each of these neighbors is tested to be (re-)determined. To avoid linear dependency of arguments of determining operations, we use the fact that the constraint graph contains all the consequences of all the constraints. Namely, the constraint graph is analyzed to test whether the arguments of the determining operation are inferred as dependent. For example, if a plane can be determined by join of three points, the constraint graph is searched for a line incident to all these points. If we have a correct operation, we obtain its priority. If the object is determined first time, we just put it in the queue. Otherwise, the new priority is compared with the previous one and the object is moved close to the start of the queue if the new priority is higher. Thus, each of the scene objects becomes known only once it is determined with the maximum possible priority. Therefore, the order of reconstruction does not depend on the order of establishing constraints.

6 Formal Sketch Correction

To allow the elevation of a 3D model, a sketch should be a true perspective projection. Namely, projective consequences of all the imposed constraints should be respected. For example, projections of 3D parallel lines should have exactly one common vanishing point. Certainly, it is not a case for perspective hand drawings due to user errors and imprecisions.

The sketch is corrected by redrawing its points and lines in the order that all the incidence constraints imposed in the sketch graph would be *exactly* satisfied. The idea is to redraw the sketch from the most constrained to the less constrained points. The order is determined by the propagation of degrees of freedom (DOF). Namely, the points with the minimal DOF and their incident lines are erased from the sketch one by another. The DOF of a point is the number of its incident lines not erased yet. All the already erased lines incidents to the current erasing point depend on it, while the not erased incident lines set the constraints that the point has to satisfy. Thus, if the DOF of the point is 0, the point is considered to be *free*. On the other hand, if the DOF is more than 2, there is an overconstrained system (Fig. 1), since for any point it is impossible to exactly satisfy more than 2 incidence constraints. In this case, the reconstruction plan (Sect. 5) is searched for a 3D point that does not implicitly determine any 3D point while itself is determined without using the projection. This projection does not affect the reconstruction, thus, once it is found, it is erased (with its incident lines) from the sketch and the new point with the minimal DOF is considered.

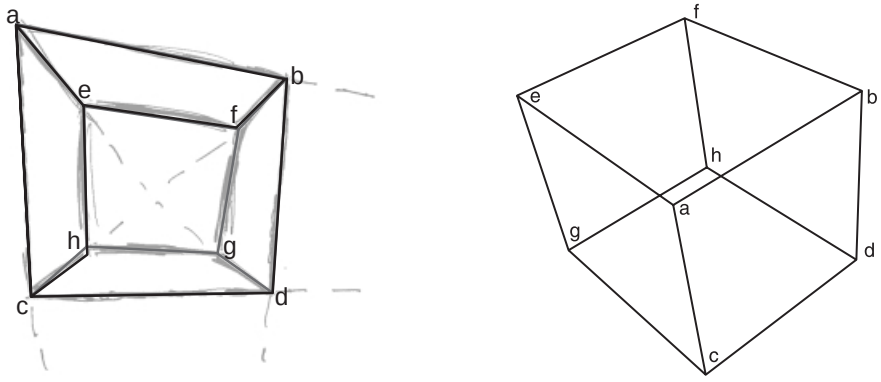

Fig. 1. The correction (left) and the reconstruction (right) of an overconstrained system. The projection of the point g is redundant, thus, it does not participate in the correction (black lines)

Once all the sketch elements are erased, they are redrawn in the *inverse* order and the formal correction plan is constructed. For each of the sketch elements (except free points), the plan contains coordinate-free expression determining the element so that it would exactly satisfy all the imposed incidence constraints. Firstly, the free points are determined, since they were most constrained. Usual free points remain unchanged, while the vanishing ones are evaluated later by weighted least squares (Sect. 7). Any sketch line is determined as a join of two firstly determined points that the line depends on. Since a line may be incident to many points, a determined line may depend on a point that is not determined yet. Such a line is added to the set of lines that constrain this point. Any point is determined as the projection or the intersection of its constraining lines once all these lines are determined. If the point has more than 3 constraining lines, there is again an overconstrained system. In this case, any point with the redundant projection is erased and the correction is restarted from scratch.

7 Estimation of Vanishing Points and Lines

In a true perspective sketch, projections of 3D parallel lines should intersect in exactly one vanishing point. On the other hand, if pencils of parallel lines contain coplanar lines, their vanishing points are aligned.

For each of the pencils of 3D parallel lines, its vanishing point is estimated by finding the closest point to the projections (i.e. sketch lines) of all the pencil lines. This point minimizes the sum of weighted squared distances to these projections where each of the weights is the maximum squared distance between all the sketch points declared as incident to the corresponding sketch line. Each of the vanishing lines that has more than 2 incident vanishing points is estimated by linear regression on these points. Finally, vanishing points that are incident to the obtained vanishing lines are projected on these lines.

8 Resolution of Orthogonalities and Eye Calibration

While orthogonality is a metric relation, it is possible to satisfy certain of such constraints without performing metric calculations *in space*. Indeed, consider three pencils of parallel lines *declared* as mutually orthogonal in space. Such a triple usually corresponds to edges of a rectangular parallelepiped called a *box*. Let we build three spheres on the sides of the *vanishing triangle* built on the vanishing points of these pencils. It is easy to verify that if the intersection of these spheres is chosen as the center of projection, 3D lines of the pencils would be *indeed* orthogonal in space after the elevation. The position of this center (*eye*) is unambiguously determined by the 2D coordinates of its projection on the sketch plane (*principal point*) and the distance between the eye and this plane (*depth*). The principal point coincides with the orthocenter of the vanishing triangle [1]. Furthermore, it is easy to verify that the depth is also determined by the coordinates of vanishing points. Therefore, we can treat scenes with any number of mutually orthogonal pencils of 3D parallel lines. Namely, to satisfy simultaneously all these orthogonalities it is sufficient to correct the sketch so that all the triples of mutually orthogonal vanishing points would define the vanishing triangles with the same orthocenters and the same depths of the eye.

8.1 Oriented Boxes

Two boxes having two parallel faces are called *oriented*. Their vanishing triangles have a common *vertical* vertex while all the other *horizontal* vertices are aligned. Such a configuration appears in urban scenes, where all the "like a box" buildings have vertical walls. Once we have aligned vanishing triangles, we set the principal point in the barycenter of their orthocenters. We have proved that it is possible to *exactly* find the minimal correction of the horizontal vertices of each of the triangles so that its orthocenter would coincide with the principal point [7]. On the other hand, it is easy to verify that aligned triangles with coincident orthocenters define the same depths of the eye.

8.2 Ordinary Boxes

Two boxes are called *ordinary* if there are no any relations between their vanishing lines. Once we have ordinary vanishing triangles, we set the principal point in the barycenter of their orthocenters. Then, for each of the triangles we find a minimal correction so that its orthocenter would coincide with the principal point. Such a correction implies a system of non-linear equations. Since its numerical solution is not exact, we reject the most corrected vertex and construct a vanishing triangle on the remaining ones with the principal point as the orthocenter. Thus, we construct the unambiguous *exact* solution [7]. Then we set the common eye depth d as the medium of the depths d_k defined by the corrected triangles. To correct each of the triangles so that it would define the depth d, we scale it with the factor $s = d/d_k$ by using the orthocenter as the center of transformation.

9 Numerical Evaluation

Once we have constructed a formal solution for a sketch correction and evaluated all the vanishing points to satisfy parallelism and orthogonality constraints, we evaluate corrections of other sketch elements by using coordinate expressions for their determining operations. Free points remain unchanged.

Once we have constructed a formal reconstruction plan, calculated the position of the center of projection and corrected all the sketch points, we evaluate the Plücker coordinates of all the elementary 3D objects composing the scene. We firstly set the eye to the obtained position and immerse it and the sketch into 3D projective space. Then the user is demanded to set in space all the points that were required to be known during the formal reconstruction (Sect. 5). The user may also set additional points. To set a point in space it is sufficient to set, for instance, its depth. Then, each of the scene objects is evaluated by applying a coordinate expression for its determining GC algebra operation. The result of evaluation of any object is invalid if it is a null tensor or represents an object at infinity while should represent the usual one. For example, a point is evaluated as infinite if its 4th coordinate is close to zero. Thus, if a result of evaluation of any object is invalid, we attempt to re-evaluate the object via all the combinations of its alternative evaluators presented in the alternative graph (Sect. 5). If the number of the alternative results close to each other is more than a certain threshold, there is just an error of precision and one of the results is returned as correct. Otherwise, there is a contradiction and the reconstruction is canceled.

It is possible to rapidly obtain reconstructions of a scene when the user incrementally enriches it with new details. Indeed, if the new elements have no any relations with the old ones, the correction and reconstruction algorithms are applied only to the new isolated subgraph of the constraint graph. Otherwise, the formal reconstruction is applied to the whole graph, while the correction algorithm is applied only to the new part of the sketch and orthogonal vanishing points are corrected so that the old part of the sketch remains unchanged. Thus, it is possible to evaluate only the changed part of the reconstruction plan.

10 Implementation and Results

Our system has been implemented on PC workstations using C++ for the kernel and Java for the UI. To reconstruct a 3D model, the user sets 3D constraints by drawing directly on a sketch. To simplify this process, the system provides the set of primitives. Then, the kernel rapidly corrects the sketch and elevates the model. The figure 2 represents the example of the reconstruction. Once all the constraints were explicitly or implicitly imposed by the user and inferred by the system, the model was obtained in 2 seconds on the PIII 733 MHz workstation.

11 Conclusion

We have presented a method that allows to accurately reconstruct 3D models even from so “unreliable sources” as perspective hand-drawn sketches once 3D

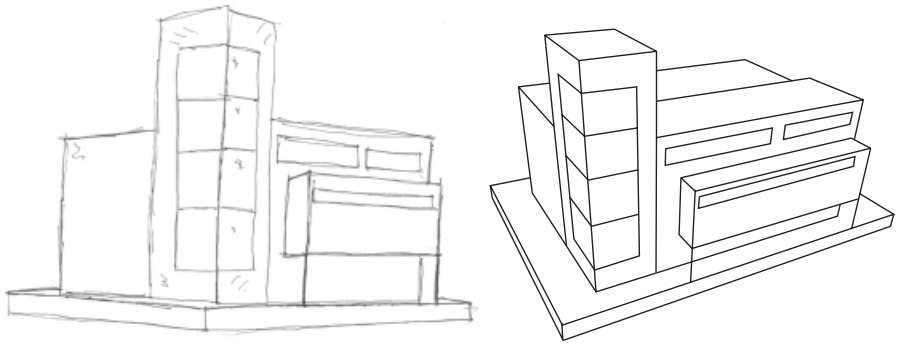

Fig. 2. The perspective hand-drawn sketch (left) and its reconstruction (right)

constraints that describe spatial structure of the models are known. The reconstruction is greatly simplified by firstly correcting the sketch and then elevating the 3D model from the obtained true perspective projection. The separation of 3D constraints and 2D preferences, the serialization of constraints and the usage of local methods to their resolution allow to avoid expensive numerical computations and thus improves the accuracy, the stability and the efficiency. The usage of projective geometric algebra allows to construct formal solutions. Having the formal solution, it is possible to compute the reconstruction in any available precision, distinguish between errors of precision and contradictions in data and rapidly reconstruct 3D scenes in the incremental manner. Future work includes the generalization of the approach to be suitable for any metric relations.

References

1. Caprile, B., Torre, V.: Using vanishing points for camera calibration. International Journal of Computer Vision, **4(2)** (1990) 127–140
2. Debevec, P.E., Taylor, C.J., and Malik, J.: Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In: Proceedings of ACM SIGGRAPH (1996) 11–20
3. Eggli, L., Hsu, Ch., Brüderlin, B., Elbert, G.: Inferring 3D models from freehand sketches and constraints. Computer-Aided Design, **29(2)** (1997) 101–112
4. Lhomme, O., Kuzo, P., and Macé, P.: A constraint-based system for 3D projective geometry. In: Brüderlin, B., Roller, D. (eds.): Geometric Constraint Solving and Applications. Springer Verlag, Berlin Heidelberg New York (1998)
5. Liebowitz, D., Criminisi, A., Zisserman, A.: Creating architectural models from images. In: Proceedings of EuroGraphics **18(3)** (1999) 39–50
6. Macé, P.: Tensorial calculus of line and plane in homogenous coordinates. Computer Networks and ISDN Systems **29** (1997) 1695–1704
7. Sosnov, A., Macé, P.: Correction d'un dessin en perspective suivant des contraintes d'orthogonalité. In: Actes de la conférence AFIG'01 (2001) 125–132
8. Ulupinar, F., Nevatia, R.: Constraints for interpretation of line drawings under perspective projection. CVGIP: Image Understanding **53(1)** (1991) 88–96