Engineering with Computers (2014) 30:31–39
DOI 10.1007/s00366-012-0283-9

ORIGINAL ARTICLE

Rapid geometric modeling for visual simulation using semi-automated reconstruction from single image

Chen Feng · Fei Deng · Vineet R. Kamat

Received: 26 March 2012 / Accepted: 6 August 2012 / Published online: 23 August 2012

Abstract This paper presents a novel algorithm that enables the semi-automatic reconstruction of human-made structures (e.g., buildings) into piecewise planar 3D models from a single image. This allows the models to be readily used in virtual or augmented reality visual simulations or for data acquisition in 3D geographic information systems. Contrary to traditional labor-intensive but accurate single view reconstruction (SVR) solutions based purely on geometric constraints, and contrary to recent fully automatic albeit low-accuracy SVR algorithms based on statistical inference, the presented method achieves a compromise between speed and accuracy, leading to less user input and acceptable visual effects. The user input required in the presented approach is primarily a line drawing that represents an outline of the building to be reconstructed. Using this input, the developed method takes advantage of a newly proposed vanishing point (VP) detection algorithm that can simultaneously estimate multiple VPs in an image. With those VPs, the normal direction of planes—which are projected onto the image plane as polygons in the line drawing—can be automatically calculated. Following this step, a linear system similar to the traditional SVR solutions can be used to achieve 3D reconstruction. Experiments that demonstrate the efficacy and visual outcome of the developed method are also described, highlighting the method's potential use for rapid geometric model building of surrounding structures in visual simulation of engineering processes.

Keywords Animation · Augmented reality · Construction · Image-based modeling · Model Engineering · Single view reconstruction · Virtual reality · Visual simulation

C. Feng (✉) · V. R. Kamat
Department of Civil and Environmental Engineering, University of Michigan, 2350 Hayward Street, Suite 2340 GGB, Ann Arbor, MI 48109-2125, USA
e-mail: cforrest@umich.edu
URL: http://www.personal.umich.edu/~cforrest/

F. Deng
School of Geodesy and Geomatics, Wuhan University, Wuhan 430079, China
e-mail: fdeng@sgg.whu.edu.cn

1 Introduction

Visual simulation allows the creation of dynamic 3D scenes using graphical computer-generated resources, and provides an environment where engineering operations, such as construction and manufacturing, can be planned and experimented with before committing real resources or endangering operational safety [1]. Visual simulation can be pursued either in virtual reality (VR) or augmented reality (AR) environments, and has emerged as a key planning and analysis tool for engineering processes in recent years [13]. To create a convincing VR or AR animation of an engineering process, detailed information about the operation and environment has to be obtained. The data must describe the underlying process, and must contain 3D models of project resources and any facility under construction, as well as of the surrounding structures and terrain. As the size and complexity of a visualized operation increase, obtaining or creating such data becomes a task that is not just arduous and impractical, but often impossible. This directly translates into the loss of financial and human resources that could otherwise be used productively.

Model Engineering is the task of acquiring, cleaning, updating, and versioning 3D models of simulation objects

for use in animation [2]. The time and effort required for Model Engineering is a significant practical challenge that deters widespread adoption of 3D visual simulation, particularly in construction practice. In an animation of a construction operation, the required 3D models include the terrain (landscape), existing structures (roads, buildings, etc.), partially complete facility, equipment (trucks, cranes, etc.), materials (steel beams, concrete blocks, etc.), and humanoids (workers). 3D models of equipment, materials, and humanoids can be obtained from 3D model vendors (e.g., www.3dcadbrowser.com, www.digimation.com). Since they are generic, the 3D models can be compiled into a library and reused in animations [16]. In addition, 3D models of facilities under construction can often be inherited from the design stages of projects [2].

The major Model Engineering problems in construction visualization are the evolving jobsite terrains and existing structures (e.g., buildings) in the vicinity of a simulated jobsite. In addition, 3D models inherited from design are often transmitted in a single or limited number of CAD layers (pieces), limiting their value in portraying a partially complete facility in animated construction. Since each project is unique in location and design, the time and effort needed to create 3D models of the local environment (terrain, existing structures, and partially complete facility) is phenomenal. For example, consider the construction of a high-rise building in an urban environment. Planners using visual simulation to study optimal crane locations on-site must obtain 3D models of not only the cranes and the planned high-rise but also of the full surrounding cityscape (buildings, roads, vehicles).

Similarly, engineers visualizing alternate plans to install a massive bridge girder with two cranes must not only acquire models of the cranes and the girder but also of the ambient landscape, roads, vehicles, and the partially completed bridge on which the girder will be placed. Such 3D models of unique jobsite landscapes, cityscapes, ambient objects, and partially completed structures cannot typically be bought or inherited; they must be manually created. Such large-scale 3D modeling for visual simulation is neither cost-effective nor practical. Model Engineering is, by definition, very demanding and time-consuming, and has been identified as one of the most significant deterrents to widespread acceptance of 3D visual simulation in scientific problem solving in general [2], and in 3D visualization of simulated construction operations in particular [23].

1.1 Research overview

Single-view reconstruction (SVR), as one of the image-based modeling (IBM) techniques, has been extensively studied from both the computer graphics and computer vision perspectives. SVR is useful when we want to recover a 3D scene with only one image at hand, which means the traditional multiple-view reconstruction approaches in either close-range photogrammetry or computer vision cannot be applied.

In the past, the main stream of SVR algorithms focused purely on the geometric structural information that one can infer from a single view of the scene as prior knowledge. The key idea of these approaches is that, through this knowledge provided by users, a scene's 3D structure can be calculated by geometry theorems. "Tour into the picture (TIP)," which was proposed by computer graphic researchers [21], is probably the earliest solution taking advantage of a vanishing point (VP) to recover 3D structures, though the assumption that the picture has only one VP limits its application. At nearly the same time that this was proposed, computer vision researchers from The University of Oxford conducted studies on single-view metrology [5, 15], introducing the theory of projective geometry that laid a solid mathematical foundation for SVR. Later, researchers from INRIA proposed an SVR algorithm based on user-provided constraints, such as coplanarity and perpendicularity, to form a linear system [17]. Compared with other similar methods [12, 20], Sturm's is regarded as one of the most flexible algorithms, and will be the basis of the SVR method in this paper.

Recently, a group of computer vision researchers has shifted its attention from geometry to machine learning to develop new SVR algorithms. Arguing that traditional SVR is labor-intensive, Hoiem et al. [13] from Carnegie Mellon University proposed a fully automatic algorithm that folds the image segments into a pop-up model. Similar algorithms include a dynamic Bayesian network SVR of an indoor image [6] and a Markov Random Field (MRF) SVR [7, 16]. Although these algorithms can achieve full automation, their reconstructed 3D model's visual effects still need to be improved for virtual or augmented reality applications.

In this research, inspired by the idea of utilizing machine learning algorithms to deduce some of the geometric structural information so as to reduce the labor burden put on users, we integrate a newly proposed line segment detector (LSD) [11] and a robust multiple structures estimator [20] into Sturm's SVR algorithm. Our method will first be compared with traditional methods in Sect. 2, then explained in detail in Sect. 3. Some of the experimental results will be given in Sect. 4, followed by a summary of this paper's contributions and a conclusion.

2 Rapid geometric modeling using SVR

2.1 Traditional SVR

Constraints A, which are provided by users, are usually parallel constraints, i.e., ones in which the image line

segments' 3D object space correspondences have the same direction. In fact, this process is equal to a manual classification of line segments—line segments whose 3D correspondences with the same direction are grouped into the same class. Ideally—when there is no measurement error—each group of line segments' extended lines should intersect at the same point in the image plane, i.e., the vanishing point. If three vanishing points are found whose corresponding 3D directions are perpendicular to one another, the camera's principle point and focal length can be calculated [2].

Constraints B are mainly coplanar constraints in Sturm's methods. By specifying which image points' 3D correspondences lie on the same 3D plane, whose normal direction is also specified through the combination of any two vanishing points, a linear system could be formed and solved. The solution of that linear system contains each image point's depth, which also means the 3D structure of the scene.

2.2 Semi-automated SVR

Compared with traditional SVR approaches, our method tries to minimize user input by taking advantage of a multiple structures estimator called j-linkage (Fig. 2).

As we can see from Fig. 2, user input constraints A and B in Fig. 1 are replaced with "user input line drawings" and "user validate/supplement constraints." This means the parallel constraints and 3D plane's normal directions will automatically be deduced, rather than specified by users. Thus, users will only need to sketch out the building to be reconstructed from a single image, leave all the computation to the algorithms, check and validate the constraints reasoned by the algorithms, and if necessary, supplement other constraints. This process will then allow the reconstructed 3D model to be viewed and manipulated on the computer.

2.3 Global assumptions

Before we explain our semi-automatic SVR algorithm in detail, there are several global assumptions that must be addressed:

No radial distortion. The image used in our algorithm should already be corrected for radial distortion, or the radial distortion parameter must be small enough to be ignored. In general, this assumption can be met easily, as long as we do not use a special lens (such as a wide-angle or fish-eye lens), and as long as the building to be reconstructed lies in the middle of the image.

The camera's principle point is located at the image center, its aspect ratio is 1, and its skew factor is 0. This assumption means that the calibration matrix of the camera has the form (with known image width W and height H, while focal length f is the only unknown parameter to be calibrated):

$$\mathbf{K} = \begin{pmatrix} f & 0 & \frac{W}{2} \\ 0 & f & \frac{H}{2} \\ 0 & 0 & 1 \end{pmatrix}. \quad (1)$$

Although the assumption that the principle point locates at the center of the image seems to be very strong, and considering the manufacturing quality of digital consumer cameras, experiments have shown that the error brought by this assumption will not induce much effect on the reconstructed model's visual effects.

A camera coordinate system is our world coordinate system. This means we ignore all six exterior parameters, i.e., the translation and rotation of the camera, so the projection matrix of the camera will take the following form: $\mathbf{P} = \mathbf{K}[\mathbf{I}|\mathbf{0}]$.

In addition, our reconstructed 3D point is up to a scale factor, meaning we are only concerned about its shape and so ignore its size. This is sufficient in many visual simulation applications, as models can be scaled if needed.

Manhattan world assumption [4]. This assumption says that a natural reference frame is given in most indoor and outdoor city scenes, as these scenes are based on a Cartesian coordinate system. Under this assumption, we could use a vanishing-point-calibration algorithm to recover the focal length of the camera.

3 Technical approach

Our proposed SVR algorithm consists of six sub-procedures, as can be seen in Fig. 2. Each module will be described in detail in the following sections.

3.1 User input

Most of the user interactions in our method are handled in this module, whose goal is to provide 2D representations of

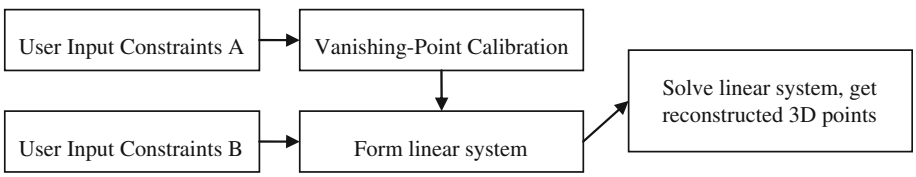


Fig. 1 General schema of traditional SVR algorithms

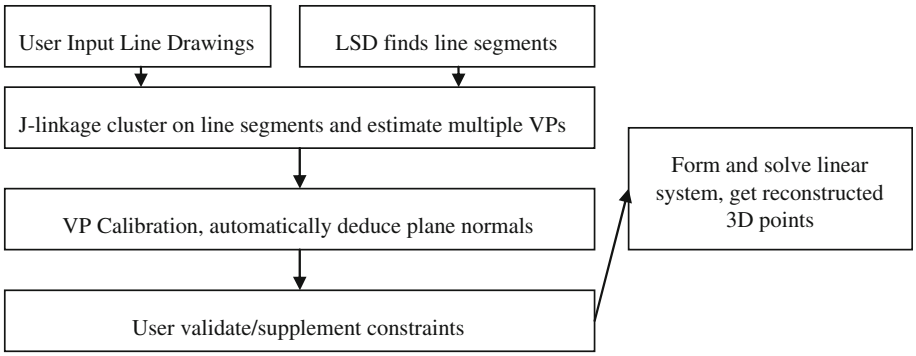


Fig. 2 Schema of our semi-automatic SVR algorithms

the target building and encoding information such as which areas of the image belong to the building, as well as where its edges and vertices are. As shown in Fig. 3, our method enables users to sketch out the skeleton of the building according to the image to be reconstructed being set as a background, with a set of line segments $\mathbf{U} = \{l_i = (x_i^s, y_i^s; x_i^e, y_i^e), i = 1, 2, \ldots, N_u\}$, in which a line segment l is represented by its two endpoints $(x^s, y^s), (x^e, y^e)$.

The data structure in this module enables the application of computational geometry algorithms to output an image point array, $\mathbf{I} = \{p_i = (x_i, y_i), \forall i \neq j, p_i \neq p_j\}$, whose elements all come from endpoints of line segments in **U**, and a set of polygons **G**, in which each polygon is represented by an ordered index array of image point array **I**.

Our SVR method's final objective will be to reconstruct the output image point array's 3D correspondence. Along with the topological information stored in polygon set **G**, one can readily recover the building's 3D model.

3.2 Line segment detector

The purpose of this module is to provide an automatic way to discover other information of the building, such as the edges of balconies, which serves an important role for further deduction. Similar to line segment set **U** in Fig. 3, LSD module's output **D** also contains line segments represented by end points. Since this set **D** will only be used as auxiliary line segments for computing the 3D correspondences of set **U**, the overlap between the two sets **U** and **D** will not affect the final 3D model.

However, different from the traditional edge-detection method that first uses the Canny edge detector followed by a series of complicated post-processing [19], the newly proposed line segment detector (LSD) [11] provides us with a fast, simple and easy-to-use interface that also gives accurate results yet requires no parameter tuning.

3.3 J-linkage

After the User Input and LSD steps, our method is ready to perform some automatic inference on the 3D structure of a building from the 2D information provided earlier. The first step is to categorize all previously acquired 2D line segments into different groups based on the direction of their 3D correspondences, which is the aim of this module. The J-linkage module contains a recently proposed robust multiple structures estimator [20], taking as input line segment sets **U** and **D** in the first two modules, and outputting sorted line segment classes $\mathbf{C} = (C_1, C_2, \ldots, C_{Nc})$, $\forall i < j, \quad |C_i| > |C_j|$, an ordered array of line segment sets sorted by their sizes, in which each element C_i is a set of line segments coming from **U** and **D** where the operator $|C_i|$ represents the number of elements of the set. Ideally, each class of line segment should correspond to a vanishing point and hence a 3D direction in object space.

The J-linkage estimator was carefully designed to robustly estimate models with multiple instances in a set of data points. This leads us to the Hough Transform [9]. However, the quantization of the parameter space—the basis of the Hough transform—will inevitably bring many shortcomings such as inaccuracy and the choice of parameterization of models. Enlightened by a popular parameter estimation approach in computer vision, RANSAC [10], and the conceptual representation from pattern recognition, the J-linkage estimator also needs no parameter tuning. Besides the estimation of multiple model instances, it could classify all data points according to the model instance they fit, which will be of significant use in our normal deduction module.

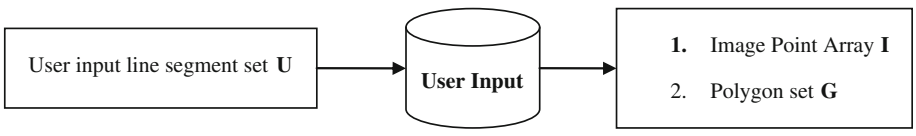


Fig. 3 Input and output (I/O) of User Input module

In our algorithm, the "data points" for J-linkage are line segments, both those drawn by users and detected by LSD, and the "model instances" are vanishing points. Line segments through different vanishing points are classified into different line segment groups. To apply the J-linkage estimator, three functions have to be defined:

1. function W that solves model parameters from a minimal number of data points,
2. function F that estimates the distance (or fitness) of a given model and a data point,
3. distance function D of a pair of data points.

For function W, one can easily figure out that the minimal number of data points (i.e., line segments) needed to solve the model parameters (the vanishing point's coordinate in the image plane) is two. Using homogeneous coordinates, it can be written as (operator $\times$ means cross product of two 3D vectors):

$$W(l_i, l_j) = \frac{l_i \times l_j}{\|l_i \times l_j\|}, \quad l_k = (x_k^s, y_k^s, 1) \times (x_k^e, y_k^e, 1), \quad k = i, j. \tag{2}$$

Function F, as the discussion in literature [19] shows, could be well approximated by the distance from the line segment's endpoint to the line connecting the vanishing point and the mid-point of the line segment (Figs. 4, 5, 6), as below (v is the homogeneous coordinate of a vanishing point; l is a line segment represented by two endpoints; dist is the distance function of the 2D point to line):

$$F(v, l) = \text{dist}(m, (x^s, y^s)), \quad m = v \times \left(\frac{x^s + x^e}{2}, \frac{y^s + y^e}{2}, 1\right). \tag{3}$$

Function D, to be used at the random sampling step in the J-linkage estimator, was not described in the literature [19]. According to the key idea of J-linkage and our experiments, function D could also be well approximated as the distance between two line segments' middle points:

$$D(l_i, l_j) = \|m_i - m_j\|, \quad m_k = \left(\frac{x_k^s + x_k^e}{2}, \frac{y_k^s + y_k^e}{2}\right), \quad k = i, j. \tag{4}$$

3.4 Vanishing-point calibration

Before carrying out other 3D information inference of the building, one piece of very important information—the focal length of the camera that generated this image—has to be computed in the vanishing-point-calibration module. In this module, based on the above-mentioned Manhattan world assumption, we further assume that the three largest line segment classes (in terms of class size) should correspond to the three coordinate basis directions in the Manhattan reference frame, which is to say their corresponding 3D directions should be perpendicular to each other. With this assumption, which is often valid in most urban outdoor and indoor scenes, there is no need for users to specify which three classes of line segments form an orthogonal coordinate system.

Therefore, the vanishing-point calibration could be automatically completed. Firstly, for each class of line segments, we apply least-square estimation to fit vanishing point through:

$$\begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ \vdots & \vdots & \vdots \\ a_n & b_n & c_n \end{bmatrix} \begin{bmatrix} v^x \\ v^y \\ v^w \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad (a_i, b_i, c_i) = \frac{(x_i^s, y_i^s, 1) \times (x_i^e, y_i^e, 1)}{\|(x_i^s, y_i^s, 1) \times (x_i^e, y_i^e, 1)\|}, \quad i = 1, 2, \ldots, n \tag{5}$$

where (v^x, v^y, v^w) is the homogeneous coordinate of the vanishing point, v, and could be solved using the singular value decomposition (SVD) algorithm. After the first two vanishing points, $\mathbf{v}_1$ and $\mathbf{v}_2$ are estimated from the two classes, the camera focal length, f, could be calculated by the equation [2]:

$$f = \sqrt{-\left(x_1 - \frac{W}{2}\right)\left(x_2 - \frac{W}{2}\right) - \left(y_1 - \frac{H}{2}\right)\left(y_2 - \frac{H}{2}\right)} \tag{6}$$

where $x_i = v_i^x / v_i^w, \quad y_i = v_i^y / v_i^w, \quad i = 1, 2.$

3.5 Normal deduction

Normal deduction is essential in many SVR algorithms [12, 18]. One of our semi-automatic SVR algorithm's

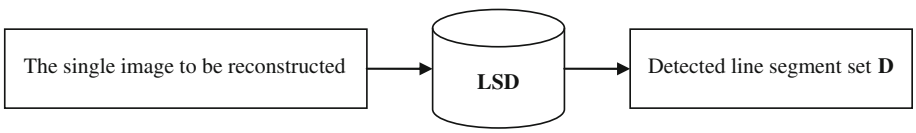
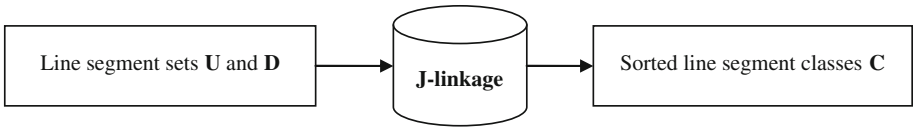


Fig. 4 I/O of LSD module

Fig. 5 I/O of J-linkage module



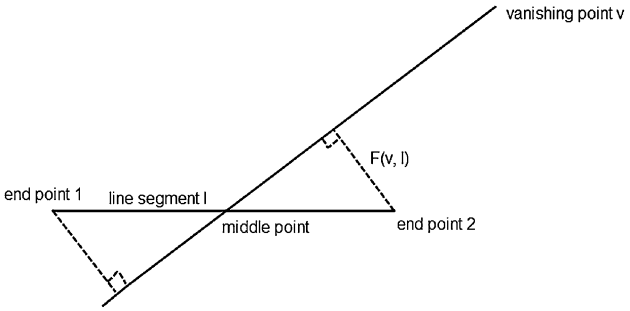


Fig. 6 Approximation of fitness function F

For each polygon **g** in polygon set **G**

From within line segment set **U** and **D**, put all line segments that lie within **g** into a new line segment set **T**

Find two line segment classes, C_i and C_j, such that among all classes, $|C_k \cap T|, \quad k = i,j$ are the two largest

Estimate two vanishing points, $\mathbf{v}_1$ and $\mathbf{v}_2$, from C_i and C_j with (5)

Calculate the unit normal direction **n** with (7), which is the deduced normal for polygon **g**

features is, in this module, automatic calculation and assignment of normal directions for each of the 3D planes that are projected onto the image plane as polygons. In traditional SVR algorithms, normal directions have to be manually specified by the user (Fig. 7).

As stated in previous sections, our method will output a piecewise planar structure as the 3D representation of a reconstructed building. Naturally, each plane in this representation has its own 3D normal direction to be deduced in this module, as shown in Fig. 8. The basic idea of this module is the fact that with two known 3D directions parallel to a 3D plane, the normal direction of the plane could be calculated, i.e., their cross product. While in camera geometry, 3D directions correspond to vanishing points in the image plane, thus one can get the following equation [16]:

$$\mathbf{n} = \frac{\mathbf{K}^{\mathbf{T}}\mathbf{l}}{\left\|\mathbf{K}^{\mathbf{T}}\mathbf{l}\right\|}, \mathbf{l} = \mathbf{v_1} \times \mathbf{v_2} \tag{7}$$

where **n** is the unit normal direction, **K** is the camera calibration matrix from (1) and (6), and $\mathbf{v}_1$ and $\mathbf{v}_2$ are homogeneous coordinates of two different vanishing points whose corresponding 3D directions are parallel to the plane with normal **n** (Fig. 9).

Once we know how to calculate the normal direction from the 3D plane's two different vanishing points, the only computation remaining is how to automatically find two vanishing points of a 3D plane (projected onto the image plane as a polygon). With the help of some basic computational geometry algorithms and the assumption that each 3D plane has several parallel lines with at least two different directions, our normal deduction algorithm could be described in the following pseudo-code:

Certainly, these assumptions do not hold in some special cases, so errors may occur and some of the calculated normal directions may go incorrect. That is why a validation step is needed for users to check those errors (Fig. 2), and this is still much easier than the traditional method.

3.6 Sturm's SVR

Finally, SVR can be performed with all crucial information obtained in the previous steps. The performing of SVR will result in the final goal of our method: 3D vertices. This module is basically the same as Sturm's SVR algorithm [18]. However, we add another kind of constraint into their original linear system: the parallelogram constraint. The purpose of this additional constraint is to make our algorithm more flexible, as parallelograms are easy to find on buildings and there is no need to use normal information when adding this constraint into the system.

The parallelogram constraint is based on the geometry fact that if four 3D points, $\mathbf{P_1}, \mathbf{P_2}, \mathbf{P_3}, \mathbf{P_4}$, could successively form a parallelogram, they must satisfy the equation

$$\mathbf{P_1} + \mathbf{P_3} = \mathbf{P_2} + \mathbf{P_4}. \tag{8}$$

Using the same parameterization as Sturm's method, if there are N image points to be reconstructed, M polygons, and K parallelograms, the linear system should be:

$$\begin{pmatrix} \underset{M\times M}{\mathbf{D}} & \underset{M\times N}{\mathbf{C}} \\ \underset{N\times M}{\mathbf{C}^T} & \underset{N\times N}{\mathbf{L}} \\ \underset{3K\times M}{\mathbf{0}} & \underset{3K\times N}{\mathbf{E}} \end{pmatrix} \begin{pmatrix} \underset{M\times 1}{\mathbf{d}} \\ \underset{N\times 1}{\boldsymbol{\lambda}} \end{pmatrix} = \begin{pmatrix} \underset{M\times 1}{\mathbf{0}} \\ \underset{N\times 1}{\mathbf{0}} \\ \underset{3K\times 1}{\mathbf{0}} \end{pmatrix} \tag{9}$$

in which $\mathbf{E}\lambda = \mathbf{0}$ expresses the parallelogram constraint, and matrices **D**, **C,** and **L** have the same meaning as Sturm's method.

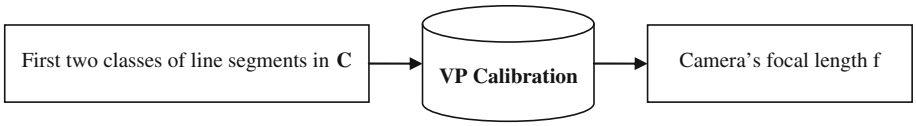


Fig. 7 I/O of vanishing-point calibration module

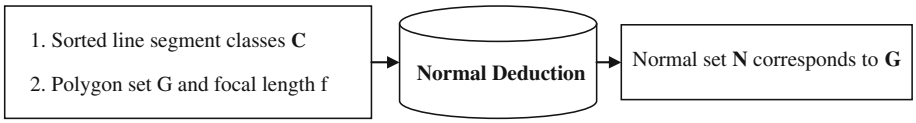


Fig. 8 I/O of normal deduction module

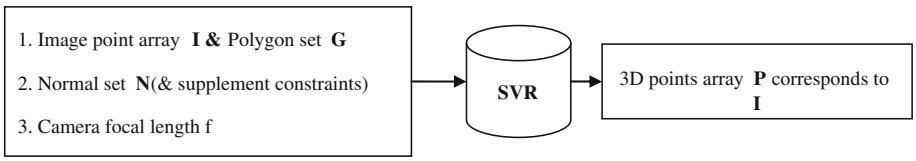


Fig. 9 I/O of SVR module

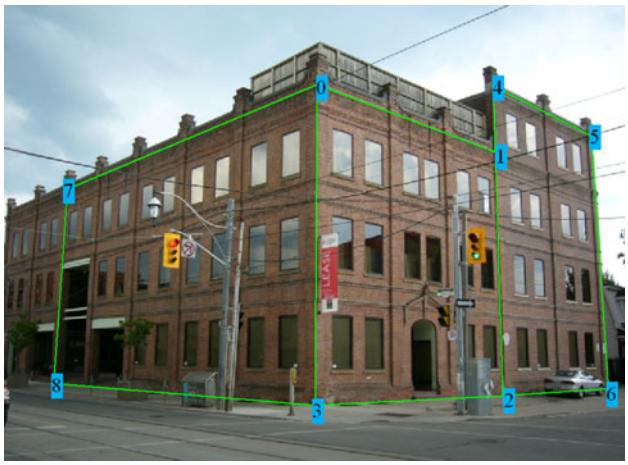


Fig. 10 User inputs a set of line segments (drawn in *green*) [7] (color figure online)

Fig. 11 LSD and J-linkage output—the first three classes are drawn in *red*, *green*, and *purple*, respectively (color figure online)

4 Experimental results

We implement the above semi-automatic SVR algorithm in the Windows XP platform using C++. The LSD module is provided by its author at http://www.ipol.im/pub/algo/gjmr_line_segment_detector/. The original J-linkage module is also provided by its author at the following website http://www.toldo.info/roberto/?page_id=46. The experimental results are detailed below.

Figure 10 demonstrates an example of how our method works. In the very first step, the image is displayed as a background. The user can draw three quadrilaterals indicating how this building looks in 2D. The three quadrilaterals are 0–1–2–3–0, 1–4–5–6–2–1, and 0–3–8–7–0 where a number 0 through 8 represents each of nine different 2D points, and the green line segments joining two 2D points are the edges of quadrilaterals. This figure shows how the intermediate result looks after the first module, User Input, is finished. Notice that the user has a large degree of freedom when choosing which part he/she wants or does not want in the final 3D result, since only regions within these three quadrilaterals will be shown in the reconstructed 3D building.

Figure 11 illustrates the intermediate result after performing the LSD and J-linkage steps. Firstly, all of the 2D line segments, either acquired from User Input—such as line segments 0–1, 5–6, or 3–8—or detected automatically by the LSD module, are sent together to the J-linkage module. After those line segments are classified by the direction of their 3D correspondences in the J-linkage module, they are shown on the image, and different groups are drawn by different colors. In this case, green line segments represent the *Y* direction, purple ones represent the *X* direction, and red the *Z* direction, which are the three

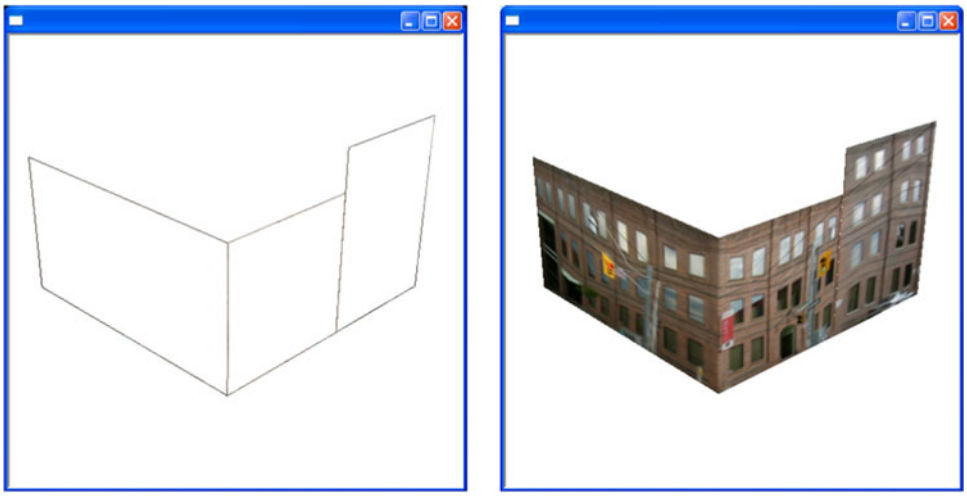


Fig. 12 Reconstructed 3D model in wire-frame and surface mode

Fig. 13 Reconstructed 3D model with curved surfaces

coordinate base directions of the Manhattan reference system. In addition, one can clearly see how the Manhattan assumption holds true here.

After all line segments are classified, our method performs both vanishing-point calibration—getting the camera's focal length—and normal deduction. As shown in Fig. 11, line segments that lie within quadrilateral 0–1–2–3–0 mostly belong to either the red (Z) or purple (X) group, which suggests this quadrilateral's 3D correspondent quadrilateral lies on the X–Z plane and its normal direction is parallel to the Z direction in the Manhattan reference system. Similarly, quadrilaterals 1–4–5–6–2–1 and 0–3–8–7–0 have normal directions parallel to the Z and X directions, respectively. Note that this procedure is achieved automatically in the Normal Deduction step in our method.

Figure 12 shows the reconstructed 3D building in both wire-frame and surface mode from a perspective different from the original image. It can be seen that the three 2D quadrilaterals are now 3D, since their 3D vertices are reconstructed by the SVR algorithm.

Figure 13 shows another challenging example reconstructed by our method. This is a difficult reconstruction task, since the building consists of curved surfaces, instead of the usual planar structures. However, our method can handle the case very well, since a piecewise planar procedure is adopted and the parallelogram constraints play an important role here, which the original Sturm's SVR method does not offer. First, the user could notice that, even though the main wall of the building is a curved surface, its two vertical edges are parallel to each other, as

are its two horizontal edges. Thus, this surface could be piecewise approximated by several parallelograms from top to bottom. This recognition is represented during the User Input step by drawing several 2D quadrilaterals corresponding to those 3D parallelograms (the yellow line segments in Fig. 13 represent the parallelogram constraints, and the blue line segments show the coplanar constraints). Finally, during the modified Sturm's SVR algorithm, these parallelogram constraints are formed into the **E** matrix in (9), benefitting from the fact that planes constrained by a parallelogram do not require the assignment of normal directions.

5 Conclusions

This paper presented a novel SVR algorithm. By utilizing a newly proposed LSD and a robust multiple structures estimator, we introduced automatic vanishing-point calibration and 3D plane normal deduction into the algorithm, thereby reducing much of the user-interaction burden. In addition, we extended the traditional SVR algorithm by adding a parallelogram as a new kind of constraint, which does not need normal direction to form the SVR linear system. In the future, we plan to consider additional approaches to make this automation more robust.

The main contributions of this paper are summarized as follows.

- By introducing LSD and J-linkage algorithms into SVR, under certain assumptions, the automation of vanishing-point calibration and 3D plane normal deduction are made possible.
- By taking advantage of a new kind of constraint—the parallelogram—and integrating it into the Sturm's SVR linear system, our SVR algorithm becomes more flexible.

Our semi-automatic SVR algorithm provides the possibility of quickly creating a surrounding human-made environment with realistic and convincing visual effects for applications such as construction visual simulation and architectural visualization. Meanwhile, its trade-off between a full automation of modeling procedure and the accuracy of reconstructed 3D results makes it suitable for rapidly populating a large 3D database, such as Google warehouse, for future applications.

References

1. Behzadan AH, Kamat VR (2009) Automated generation of operations level construction animations in outdoor augmented reality (special issue on Graphical 3D Visualization in Architecture, Engineering, and Construction, American Society of Civil Engineers, Reston, VA). J Comput Civil Eng 23(6):405–4417
2. Brooks FP (1999) What's real about virtual reality? IEEE Comput. Graph. Appl 19(6):16–27
3. Caprile B, Torre V (1990) Using vanishing points for camera calibration. Int J Comput Vision 4:127–139
4. Coughlan JM, Yuille AL (1999) Manhattan world: compass direction from a single image by Bayesian Inference. In: International conference on computer vision (ICCV), 1999
5. Criminisi A, Reid I, Zisserman A (2000) Single view metrology. Int J Comput Vision 40:123–148
6. Delage E, Lee H, Ng A (2006) A dynamic bayesian network model for autonomous 3d reconstruction from a single indoor image. Computer vision and pattern recognition (CVPR) 2:2418–2428
7. Delage E, Lee H, Ng A (2007) Automatic single-image 3d reconstructions of indoor manhattan world scenes. In: Robotics Research, vol 28, pp 305–321. ISBN: 978-3-540-48110-2
8. Denis P, Elder J, Estrada F (2008) Efficient edge-based methods for estimating manhattan frames in urban imagery. In: European conference on computer vision, 2008
9. Duda RO, Hart EP (1972) Use of the Hough transformation to detect lines and curves in pictures. Commun ACM 15(1):11–15
10. Fischler MA, Bolles RC (1981) Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun ACM 24:381–395
11. Grompone G, Jakubowicz J, Morel J, Randall G (2010) LSD: a fast line segment detector with a false detection control. IEEE Trans Pattern Anal Mach Intell 32:722
12. Grossmann E, Santos-Victor J (2005) Least-squares 3D reconstruction from one or more views and geometric clues. Comput Vis Image Underst 99:151–174
13. Hoiem D, Efros AA, Hebert M (2005) Automatic photo pop-up. ACM Trans Graph (TOG) 24:584
14. Kamat VR, Martinez JC, Fischer M, Golparvar-Fard M, Pena-Mora F, Savarese S (2011) Research in visualization techniques for field construction (American Society of Civil Engineers, Reston, VA). J Constr Eng Manage 137(10):853–862
15. Liebowitz D, Zisserman A (1999) Combining scene and auto-calibration constraints. In: The 7th international conference on computer vision, Kerkyra, Greece 1999
16. Min P, Halderman JA, Kazhdan M, Funkhouser TA (2003) Early experiences with a 3D model search engine. In: Proceeding of the eighth international conference on 3D Web technology, ACM Press, New York, pp 7–18
17. Saxena A, Chung S, Ng A (2006) Learning depth from single monocular images. Adv Neural Inf Process Syst 18:1161
18. Sturm P, Maybank SJ (1999) A method for interactive 3d reconstruction of piecewise planar objects from single images. In: British machine vision conference, Nottingham, England, 1999
19. Tardif JP (2009) Non-iterative approach for fast and accurate vanishing point detection. In: International conference on computer vision (ICCV), 2009
20. Toldo R, Fusiello A (2008). Robust multiple structures estimation with J-Linkage. In: European conference on computer vision (ECCV), 2008
21. Van Den Heuvel FA (1998) 3D reconstruction from a single image using geometric constraints. ISPRS J Photogramm Remote Sens 53:354–368
22. Youichi H, Ken-Ichi A, Kiyoshi A (1997) Tour into the picture: using a spidery mesh interface to make animation from a single image. In: Proceedings of the 24th annual conference on computer graphics and interactive techniques
23. Zhang H, Shi JJ, Tam CM (2002) Iconic animation for activity-based construction simulation. J Com Civil Engin 16(3): 157-164