

## Co-Ordinate: Reconstruction of Three Dimensional Geometric Diagrams from a Single Image

Katsutsugu Matsuyama    Kouichi Konno

Iwate University  
kmatsu @ iwate-u.ac.jp

### Abstract

We designed a new algorithm for reconstructing three-dimensional (3D) geometric diagrams. By extracting the orthogonality of the coordinate axes, the contents of labels, the positional relationships between labels and other elements, and direct and indirect connection relationships between elements, we define a score function and derive the optimal parameters that minimize it. This allows the elements to be reconstructed in 3D space and allows the system to create diagrams with different viewpoints. In our system, the reconstructed elements are redrawn with explicit orthogonal coordinate axes. The user can directly move the axes as an interface for changing viewpoints. The system allows the user to edit and attach elements to the diagram interactively.

## 1 Introduction

Diagrams depicting three-dimensional (3D) geometric information, as shown in Figure 1, are frequently used as a means of communicating information that includes geometrical concepts, as well as composition and structure, in fields such as mathematics, engineering and design.

In this paper, we propose a novel technique for reconstructing 3D geometric diagrams from a single image. We define a “3D geometric diagram” as a diagram in which (1) geometric information in 3D space is expressed by line segments, points, and labels, (2) 3D coordinate axes are represented by line segments or arrows, and (3) geometric information in 3D space is drawn by parallel projection. 3D geometric diagrams can often consist of less geometric elements (such as line segments and strokes) than 3D objects because they use labels and assumptions to notate traditional 3D geometric representation.

We design a new algorithm for reconstructing 3D geometric diagrams. By extracting the orthogonality of the coordinate axes, the contents of the labels, the positional relationships between labels and other elements, and direct and indirect connection relationships between elements, we define a score function and derive the optimal parameters that minimize it. There are various methods for reconstructing 3D objects from images composed of line segments or sketches (see Section 2). These methods set the horizontal and vertical directions of the input image as  $x$  and  $y$  axes respectively and derive the depth direction ( $z$  coordinate). This can be characterized as reconstructions in screen coordinate system. On the other hand, our method identifies the coordinate axes from among the line segments in the input image and can be characterized as reconstructions in world coordinate system. We could not find existing techniques of identifying the coordinate axes. The feature of our method is employing labels to identify the coordinate axes, focusing that many of 3D geometric diagrams have labels.

The elements are then reconstructed in 3D space, which allows the system to create diagrams with different viewpoints. Our system redraws the reconstructed elements with explicit orthogonal coordinate axes. Users can directly move the axes as an interface for changing viewpoints. The system allows users to edit and attach elements to the diagram interactively with familiar 2D-like user interface.

It is anticipated that our technique will be employed in many applications, such as the reverse engineering of printed figures, improving impressions by changing figure viewpoints, and when making fair copies of hand drawn figures from the best

viewpoint. In addition, the trial-and-error process used to adjust figures can be reduced by our technique in various situations, such as cases where there is some overlap in a display when a new element is added to a diagram with existing elements (such as lines, points and text). Other situations where the technique could prove useful include trying out the visual effect of mirror image conversion or orthogonality/non-orthogonality, or when visually integrating multiple diagram coordinate systems.

## 2 Related Work

### Reconstructing 3D objects from 2D line drawing

In the fields of computer vision and computer-aided design (CAD), substantial research has been conducted over the years into methods for reconstructing 3D objects from images composed of line segments or sketches. [1, 2] explain works of the early studies for interpreting line drawings, including labeling lines to find planar surfaces and corners, interpreting curved lines, and recovering depth direction using convex/concave and T-junction etc. as constraints. Many techniques construct an energy function using the heuristic regularity of images such as parallelism, perpendicularity, and symmetry [3, 4, 5]. Refinement techniques using hidden lines to reconstruct more complex objects [6, 7, 8] and inferring the back of the object from drawings without hidden lines [9] have been also studied.

The methods of [1-9] set the horizontal and vertical directions of the input image as  $x$  and  $y$  axes respectively and derive the depth direction ( $z$  coordinate). Masry et al. [4] select the group of three line segments as tentative (object coordinate) axes to reconstruction but this is not directly applicable to the reconstruction of 3D geometric diagrams because the three lines are selected based on only 2D angles of line segments and no information about the origin in world coordinate system. Accordingly, we have referred to the technique described in [4] and have designed a new algorithm for 3D geometric diagrams that takes into consideration the characteristics of geometric diagrams.

In recent years, many studies have been conducted into various modeling interfaces. For example, [10, 11, 12, 13] have proposed sketch-based interface. These studies support creation and correction by the user of 3D models and 3D scenes, and cannot be directly applied to our purpose since our system takes a single image as input.

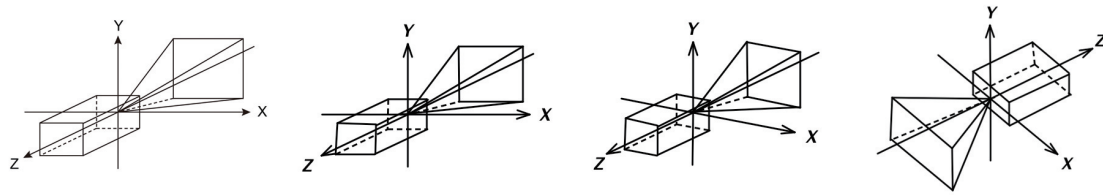


Fig. 1: An example of executing the proposed system Co-Ordinate. The diagram on the far left is an input image. The second diagram is an image redrawn by extracting elements from the input image. Users can manipulate these diagrams interactively. Since the elements were reconstructed in 3D space, users can create diagrams with different viewpoints (like those in the third and fourth position) simply by dragging the coordinate axes.

### 3D information extraction from a single image

In our proposed method, 3D information is extracted from a single 2D diagram image. Numerous studies have been conducted on extraction and recognition of objects from an image, and specific research efforts aimed at the extraction of 3D information from a single image have been introduced.

Most techniques employ specific characteristics for extraction. Employing vanishing points etc., object size calculation [14] and a modeling interface [15] are offered. Hoiem et al. created 3D pop-ups employing category segmentation [16], while Jiang et al. proposed a method for modeling symmetrical 3D buildings [17].

The above studies perform object extraction/recognition from images, focusing on photos of natural and man-made objects, and the techniques cannot be directly applied to extraction from artificial images such as diagram images.

#### Orthographic and oblique projection

We employ the (projected) orthogonal coordinate axes as an interface for changing the viewpoints of parallel projection. While parallel projection can be classified into two major types, orthographic projection and oblique projection, oblique projection has not been discussed significantly in the field of computer graphics. One reason for this is that schemas for 3D systems, such as camera rotation, cannot be used with oblique projection. Research in which oblique projection has been used includes a study representing artificial sense [18] and a study involving drawing of a map with high legibility [19]. However, these studies use oblique projection as a means of visualization, and they differ from studies of interaction techniques that focus on the oblique projection diagrams themselves.

A number of studies have previously been carried out regarding the creation of new projections. These include nonlinear projections that can be manipulated interactively [20], techniques combin-

ing multiple views along a camera path [21, 22, 23], studies proposing new camera models [24, 25], and studies that make pseudo changes in projections by warping the image [26]. However, in these cases, it is hard to achieve the purpose of this research, which is intuitive manipulation of oblique projection parameters. The proposed system targets both orthographic and oblique projection and provides an interface for manipulating viewpoints.

There are several interface techniques that can be used to change viewpoints using widgets [27, 28], these techniques are designed for the context of 3D perspective scenery.

## 3 Reconstructing 3D information from an image

Many existing diagram images depicting 3D geometric information were actually drawn with 2D drawing software, and then stored in bitmap format. Our system takes a bitmap image as input, reads off diagram elements from the image, and then reconstructs the 3D geometric elements.

As indicated in Fig. 2, the reconstruction system proposed in this paper is composed of four modules: label extraction, 2D element extraction, coordinate axes identification, and 3D reconstruction. The label extraction module reads off characters from the input image, and then outputs the results (a table of characters and bounds) along with the image (without labels). The 2D element extraction module extracts 2D geometrical elements (e.g., line segments and points) from the image input to the module, identifies the elements to which the label belongs, and then saves the results as 2D data. The coordinate axes identification module identifies the coordinate axes from among the line segments in the image based on the 2D and label data obtained in the preceding process, and then saves the results as coordinate axes information. The 3D reconstruction module reconstructs the 3D geometry

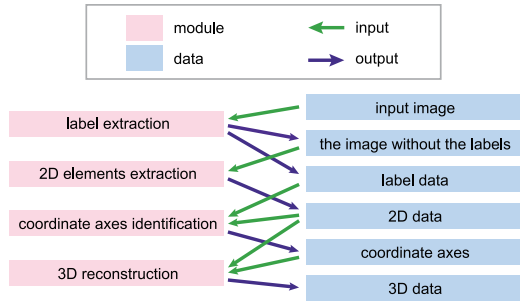


Fig. 2: Overview of our reconstruction system

from 2D data based on the identified coordinate axes.

### 3.1 Label extraction

Numerous diagram images contain labels such as '0' to indicate the origin, 'X' to indicate a coordinate axis, and 'P' to indicate a point label. We designed our system to utilize labels for the following processes. Our label extraction module first performs binarizing and four-neighbor labeling, after which Tesseract optical character reader (OCR) software is used to scan and analyze each label.

If the system passes over a label or makes reading mistakes, the user would add or correct the characters by dragging and inputting text. The user would also apply more sophisticated OCR applications, including commercially available software packages.

The label extraction module outputs the results of the read text, along with the image with the labels removed. Fig. 3 shows an input/output example of the label extraction module.

### 3.2 2D elements extraction

The 2D element extraction module extracts 2D geometrical elements from the image input to the module, identifies the elements to which the label belongs, and then stores the results as 2D data. While the system allows the user to create and specify 2D data, we implemented the function used to read off segments and points from the input image in order to automate and facilitate such tasks. Segments are differentiated into solid lines and dotted lines.

Both the Hilditch algorithm (for line thinning) and the Hough transform (for detecting lines) are applied. The position of the end of line segment,

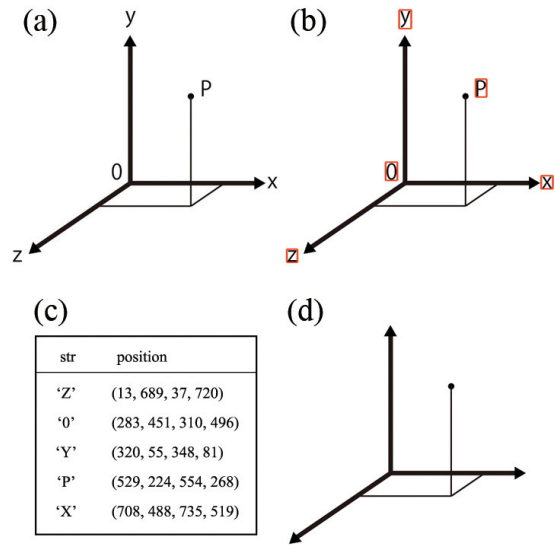


Fig. 3: Example of input/output of label extraction module. (a) Input image. (b) Applying OCR to the input image. The system outputs the reading results (c) and the image with the label removed (d).

along with the line thickness and type, are calculated from both the detected line data and the input image. The position of the end points are determined by searching for continuous (with 16-pixel looseness) pixels from the set of pixels on the detected lines. The thickness of the line segment is found by searching for pixels in the direction perpendicular to the line segment. The line type is determined to be dotted if the percentage of black pixels in the line segment is less than 70 %, and solid line if the percentage is 70 % or more.

Next, the points, actually filled-in circles, are extracted. The system searches for filled-in circles present at the end points of the line segments. In particular, the system creates a circle with its center on the extracted line segment and a diameter 1.5 times the thickness of the line segment, so that it passes through the end point of the line segment. If the percentage of black pixels in the circle is greater than a threshold value (95 %, in our study), it is then determined that there is a point at the end point of the line segment. From the extracted 2D elements, the system selects one as the element to which the label belongs. Our algorithm first searches for the point closest to the label, and then checks whether or not the distance is less than the threshold value. If such a point cannot be found, the system searches for the nearest line segment.

The user can correct the extraction result if necessary. There are numerous segment vectorization methods [29, 30, 5]. By combining our algorithm

with one of these methods, detectability, and robustness would be improved.

### 3.3 Identification of coordinate axes

Most 3D geometric diagrams depict the coordinate axes explicitly or implicitly in order to provide spatial cues for human readers. In our method, the coordinate axes identification module selects a combination of three line segments as the orthogonal coordinate axes from a set of input line segments. Using actual 3D geometric diagram images, we analyzed the characteristics of coordinate axes and designed the algorithm described below for identification use.

The candidates for coordinate axes are the solid line segments (e.g., extracted in Section 3.2). Our method calculates a score ( $S_{seg}$ ) for each line segment that indicates the degree to which the line segment has the features of a coordinate axis. The  $S_{seg}$  is calculated as follows:

$$S_{seg} = S_{len} + S_{diam} + S_{xyz} + S_o + S_{freq} \quad (1)$$

$S_{len}$ : The longer the line segment, the higher the possibility that it is a coordinate axis. We set  $S_{len} = \text{line segment length} * \alpha_0$ , where  $\alpha_0$  is a weight coefficient. In the following terms section, weight coefficients are used as a multiplier, just as in this term.

$S_{diam}$ : The thicker the line segment, the higher the possibility that it is a coordinate axis. The line segment thickness obtained in Section 3.2 is used here.

$S_{xyz}$ : The shorter the distance between the line segment and the labels X, Y or Z (if any), the higher the possibility that it is a coordinate axis ( $S_{xyz} = \text{distance}^{-1}$ ). Based on the results of label extraction in Section 3.1, the system employs the shortest distance between the line segment and the X, Y or Z labels. The shorter the distance, the higher the value.

$S_o$ : The shorter the distance between the line segment and the label 0, the higher the possibility that it is a coordinate axis. The shorter the distance between the line segment and the label 0, the higher the value.

$S_{freq}$ : Based on the assumption that diagrams have numerous line segments drawn parallel to the coordinate axes, we then create a histogram of the 2D angles of those line segments by applying a Gabor filter [31] with 1 degree intervals and summing the pixel values. We created a Gabor kernel using the `getGaborKernel` function in OpenCV [32]. The parameters of the Gabor kernel are  $\sigma=8.0$ ,  $\lambda=10.0$ ,

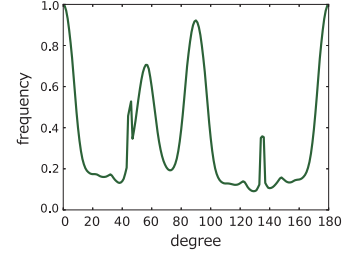


Fig. 4: Histogram indicating the slope of line segments. The input image is Fig. 3. In this graph,  $\theta=0$ [degree] indicates the vertical direction, while  $\theta=90$ [degree] indicates the horizontal direction.

$\gamma=0.5$ , and  $\phi=0$  (for more details about each parameter, see the OpenCV documentation).

Fig. 4 shows an example of the results obtained when Fig. 3 was used as the input image, which is a histogram indicating a slope of line segments. This histogram is roughly equivalent to that obtained via Angular Distribution Graph (ADG) in [4]. However, it should be noted that ADG takes vectors as input, while our method takes the image itself.

In determining coordinate axes, the combination of the three line segments that maximize the following score  $S_{axes}$  is identified as the coordinate axes set:

$$S_{axes} = S_{sum} + S_{dist} + S_{thick} \quad (2)$$

$S_{sum}$ : Sum of  $S_{seg}$ s for three line segments.

$S_{dist}$ : The shorter the distance between line segments, the higher the value.

$S_{thick}$ : Differences in line segment thickness are taken into account as well. The smaller the thickness differences, the higher the value.

If the labels X, Y and Z exist, the system assigns  $x$ ,  $y$  and  $z$ , along with positive directions, to the line segments based on the distances between the line segments and the characters.

[4] also selects the group of three line segments based on a discrete histogram of the 2D angles of line segments. The selected group consists of the tentative axes used locally to reconstruction. [4] basically considered angles alone, and is, therefore, not applicable to our case.

### 3.4 3D reconstruction

The 3D reconstruction module reconstructs a 3D geometric diagram from the 2D data based on the identified coordinate axes. The projection parameters are calculated using the line segments for the coordinate axes.

The position  $(x_0, y_0)$  of the 3D origin is placed on the canvas is the position where  $(0, 0, 0)$  is pro-

jected, and the vector  $(a, b)$ , where the  $x$  axis are placed on the canvas, are regarded as the vectors where  $(s_x, 0, 0)$  is projected. The vectors  $(c, d)$  and  $(e, f)$  are also regarded that  $y$  axis  $(0, s_y, 0)$  and  $z$  axis  $(0, 0, s_z)$  are projected respectively.

Here,  $s_x$ ,  $s_y$  and  $s_z$  indicate the scales of the projection axes.  $s_x$ ,  $s_y$  and  $s_z$  are values that can be changed by the user (Section 4), and which are initially set to 1.0. The projection of the 3D space  $(X, Y, Z)$  onto the canvas space  $(x, y)$  is given by the following formula:

$$(x, y) = (a, b)X/s_x + (c, d)Y/s_y + (e, f)Z/s_z + (x_0, y_0) \quad (3)$$

where  $(a, b)$ ,  $(c, d)$  and  $(e, f)$  are the tip of the  $x$ ,  $y$  and  $z$  axis on the canvas respectively. Conversely, the canvas space  $(x, y)$  can be converted to projection rays in 3D space by using the parameter  $t$  and the following equation:

$$(X, Y, Z) = (x - x_0, y - y_0, 0) + (cf - de, eb - fa, ad - bc)t \quad (4)$$

Reconstructing 3D geometric information corresponds to determining the proper parameter  $t$  for each element comprising the diagram. Since the line segments are the key to the reconstruction, and since each line segment has two end points, it is necessary to find the proper parameter  $t$  for each end point.

Factors such as the position and the connection relationship between line segments can be extracted from images. Therefore, we take the approach of using these, defining a score  $T$  for relevance, and then searching for a parameter  $t$  combination that yields an optimal score. The score  $T$  is calculated as follows:

$$T = T_{parallel} + T_{direct} + T_{axes} + T_{indirect} \quad (5)$$

We then employ simulated annealing [33] to search for the parameter  $t$  combination that minimizes the score  $T$ . Each term is as follows:

### Parallelism of line segments

Parallel line segments are evaluated so that these segments are parallel in 3D space as well. At initialization, the system collects parallel line segments, including the coordinate axes. During evaluation, the angle between the line segments is calculated as a penalty. Just as in Section 3.3, each term is multiplied by a weight coefficient ( $\beta_0$ ).

### Direct connection relationships

Direct connection relationship means a case where two line segments connect at common point. For

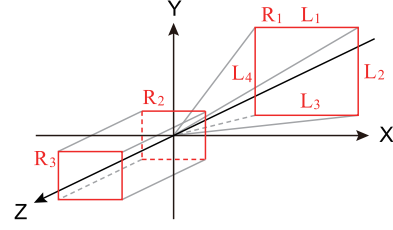


Fig. 5: An example of direct and indirect connection relationships.

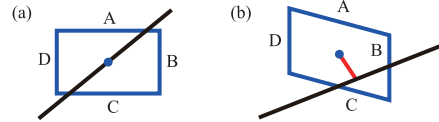


Fig. 6: Indirect connection relationships. (a) Our system determines that a relationship exists if the geometrical center of gravity of the segments A-D (blue) lies on a certain segment. (b) The distance between the center of gravity and the segment is calculated as a penalty.

example, two line segments  $L_1$  and  $L_2$  in Fig. 5 have a direct connection relationship. Evaluation is performed to achieve the shortest distance (in 3D space) between each pair of line segments with a connection relationship. At initialization, the system constructs a network graph to indicate the connection relationships between line segments. All of the line segments are registered in the graph as nodes, and line segments that have a connection relationship are linked. This link is equivalent to an edge in terms of graph theory. Determining whether line segments A and B have a connection relationship is performed by (1) determining whether the distance between line segment A and either end point of line segment B is at or below the threshold value, or (2) determining whether the distance between line segment B and either end point of line segment A is at or below the threshold value. In this paper, the threshold value is set to 12 [pixel]. The graph is an undirected graph. During the evaluation process, the system searches for a pair of nodes connected by an edge in the network graph, after which the sum of the minimal distances between line segments in 3D space is calculated as a penalty.

### Axes segments

Distance indicating how far the position in 3D space of the segment identified as the coordinate axis is shifted from the actual coordinate axis. In the case of the  $x$  axis, using two end points  $(x,$



$y, z$ ) of the line segment,  $\sqrt{y^2 + z^2}$  is used as the evaluation value.

### Indirect connection relationships

Indirect connection relationship means a case where the center of a shape is on a line segment. Evaluation is then performed while taking into account indirect connection relationships to ensure that these connections are maintained. At the term  $T_{direct}$ , the direct connection relationships are evaluated. However, it is possible for a line segment to pass through the center of a shape consisting of multiple line segments without having a direct connection relationship between those line segments, even though a relationship between the overall shape and the crossing line segment can be seen. An example of this can be seen in the bottom of the pyramid shown in Fig. 1, which is comprised of line segments (the shape), and the line segment that passes through the center of that rectangle. Rectangles (a group of four line segments)  $R_1, R_2$  and  $R_3$  in Fig. 5 are also examples of indirect connection relationships. Each rectangle has an indirect connection relationship with  $z$  axis as the center of each rectangle is on the  $z$  axis.

We then implement the automatic search function to detect indirect connection relationships. In this process, the system extracts the cycles (graph theory term) from the network graph constructed above. Among these, the applicable cycles are taken to be those in which a line segment identified as a coordinate axis is not included in a node, and those in which the number of elements is four or more. The geometrical center of gravity is calculated for the figure composed of the nodes of each applicable cycle. If there is a line segment touching the geometrical center of gravity of the cycle, a relationship is determined to exist (Fig. 6 (a)). The user can switch the relationship of each indirect connection relationship on or off as desired.

During evaluation, the system calculates the distance between the geometrical center of gravity of the cycle in 3D space and the line segment that it passes through as a penalty (Fig. 6 (b)).

## 4 User Interface

In this section, we will describe our user interface for manipulating a 3D geometric diagram. As shown in Fig. 7, our user interface resembles conventional drawing tools. The top-level layout of the user interface consists of a tool panel, canvas, and a data table. The user selects a desired element from the tool panel, and places it on the canvas. He or

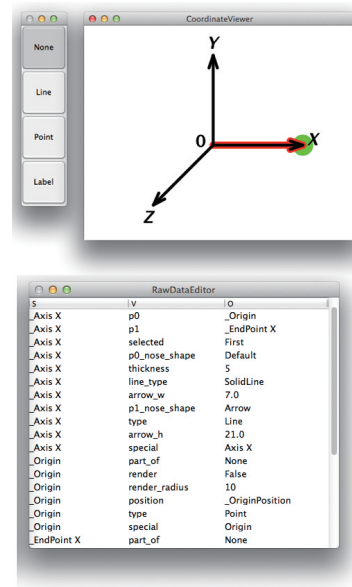


Fig. 7: Top-level layout of user interface.

she can then edit elements on the canvas, as well as correct data directly through the data table. The coordinate axes are displayed in the form of arrows on the canvas. After reconstructing the 3D information from the diagram image, the extracted information is also redrawn with explicit coordinate axes. The user can interact with the diagram by directly moving the axes to change viewpoints.

The system allows the user to edit and attach elements to the diagram interactively with 2D-like user interface that allows him or her to create and edit points, lines and labels. The following sections describe the specific user interface for each of these elements.

To ensure that all elements maintain their correct positions even if the coordinate axes are moved, all elements must be associated with their 3D positions. In our method, each point has both a 2D position  $p(x, y)$  and a 3D position  $P(x, y, z)$  for position information. When the user determines a position  $p$  in the canvas space by manipulating a point, the system calculates the proper 3D position  $P$ .

### 4.1 Coordinate axes

In the initial state, the canvas displays the arrows indicating the coordinate axes, and characters 0, X, Y and Z, which indicate the origin and coordinate axes (Fig. 7). If the user drags the area near the origin, all of the elements will be translated. When the user moves the coordinate axes, all of the elements move as well, while retaining their spatial

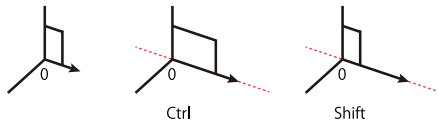


Fig. 8: Movement of coordinate axis tip.

coordination. From the coordinate axes, the system calculates projection parameters as shown in Equation (3).

If the tip of a coordinate axis is dragged while pressing the Ctrl key, movement will be performed in a fixed direction. If dragging is done while pressing the Shift key, the direction will remain fixed and only the coordinate axis will be lengthened or shortened (Fig. 8); This can be achieved by changing the scales  $s_x$ ,  $s_y$  and  $s_z$  in accordance with the movement amount.

## 4.2 Points

When creating a point at position  $p$ , the intersection  $P$  between the parallel projection rays (Equation (4)) from the position  $p$  in the canvas space and the  $zx$  plane ( $y = 0$ ) is taken to be its position in 3D space. When a point in canvas space touches a line segment already present on the canvas, the position obtained by linear interpolation between the 3D positions of both ends of the line segment is assigned to  $P$ .

## 4.3 Lines

The 3D positions of the end points are determined using the same method as described in Section 4.2. If either end point (indicated here as  $p_1$ ) snaps to an already existing point or line segment during translation of a line segment on the canvas, the 3D position of the existing element is designated as  $P_1$ .

## 4.4 Labels

Labels are characters with attachment information. Users can attach a label to any point or line segment by dragging. When a label is in the attached state, its display position on the canvas is automatically calculated so that it will be positioned close to, but not overlapping its element.

Labels do not have 3D position information, and simply use the 3D information of the attached point or line segment indirectly. When displaying labels on the canvas, the system takes the distance with

the attached element as a score, and then searches for a position where the score is low. A penalty is added to the score if there is overlap with elements other than itself in the canvas space.

## 5 Experimental Results and Discussion

We implemented the system for extracting information from an image described in Section 3, along with the interactive interface described in Section 4. The weight coefficients are  $\alpha_0 = 10^0$ ,  $\alpha_1 = 10^0$ ,  $\alpha_2 = 10^{-1}$ ,  $\alpha_3 = 10^{-1}$ ,  $\alpha_4 = 10^0$ ,  $\beta_0 = 10^5$ ,  $\beta_1 = 10^1$ ,  $\beta_2 = 10^5$ , and  $\beta_3 = 10^3$ . A triplestore database (example of good reading [34]) was used as the data format output by the information extraction system, and the input/output data format for the interactive manipulation system. The results of extracting information from the image can be passed directly to the interactive manipulation system. A triplestore database has the advantage of allowing the data to be flexibly expanded. On the other hand, it is possible to recalculate 3D position information by outputting all or part of the information from the interactive manipulation system as an image, and then re-inputting that image to the information extraction system. The user can consider the output image as a re-input image and can complete processing only by image data. The following section describes and discusses the results of executing the system for extracting information from images and the interactive manipulation system.

### 5.1 System for extracting information from an image

As shown in Fig. 1 and Fig. 9, 3D information can be reconstructed for a number of 3D geometric diagram images. Fig. 9 includes examples of printed figures images and images made by scanning and hand-drawing coordinate axes to line-drawing. Since the system builds its geometric topology at the reconstruction process, hidden line and surface removal would be possible. At the bottom of Fig. 9, although this image can actually have multiple interpretations, our algorithm output the one like the third and fourth images based on our assumptions for 3D reconstruction. We found some cases that need user correction, as shown in Fig. 10.

The system was designed so that, when calculating the score  $S_{seg}$  in Equation (1) of Section 3.3, it is possible to adjust the contribution of each term



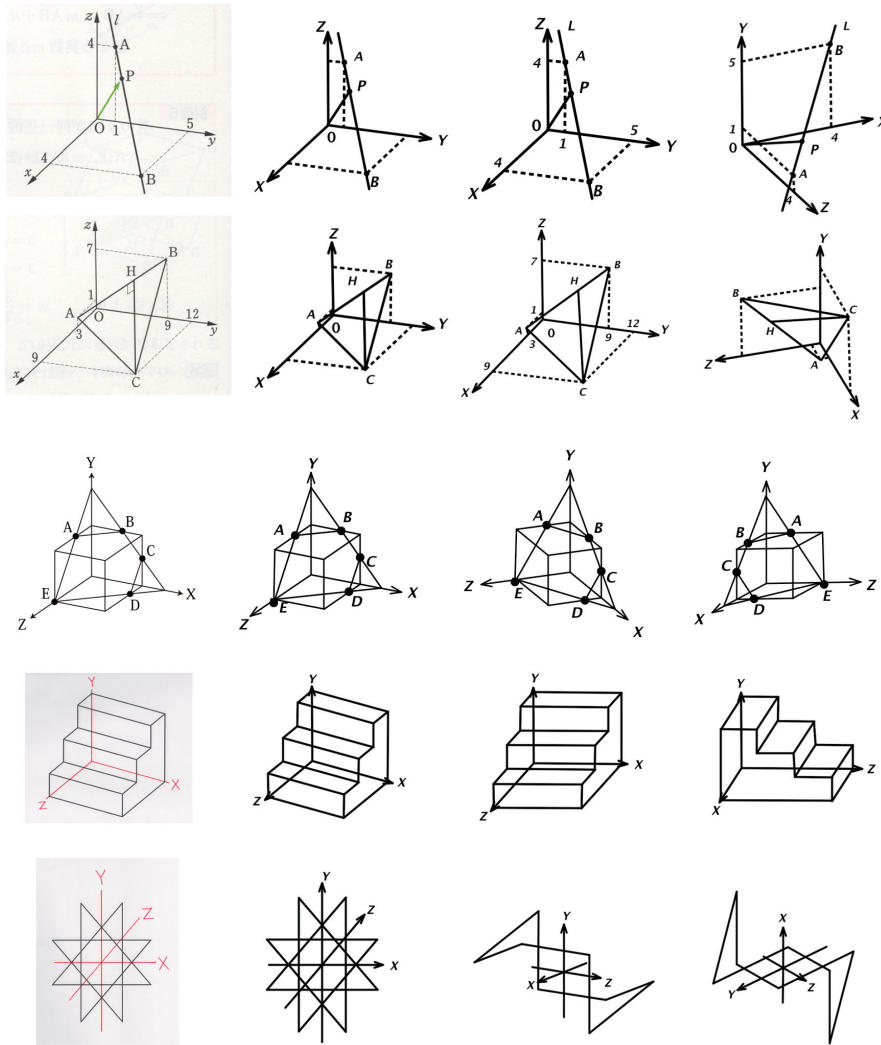


Fig. 9: Experimental results. The far left is an input image. The second is an image redrawn by extracting elements from the input image. The third and fourth are diagrams with different viewpoints.

using a weight coefficient  $\alpha$ . Thus, by trying various coefficient adjustments, the authors learned that  $S_{xyz}$ , i.e., the distance between the line segment and the characters X, Y and Z, was important when identifying coordinate axes. It can be presumed that this is due to the communications technique of promoting recognition of coordinate axes by placing arrows and characters in close proximity with each other on the diagrams.

We implemented our system using the Python programming language and executed on Mac OS X (CPU: 3.4GHz Intel Core i7, Memory: 8GB 1333MHz DDR3, GPU: AMD Radeon HD 6970M 1024MB). Table 1 shows the calculation time for processing our algorithms. The 3D reconstruction module was dominant and the iteration counts of simulated annealing were 15,000 and 4,000 times

respectively. While Intel Core i7 is multi-core CPU, we did not perform program parallelization. As a result of trying to tune parameters, the simulated annealing tends to converge successfully when weight coefficients are  $\beta_2 \geq \beta_0 > \beta_3 > \beta_1$ . Fig. 11 shows convergence of the simulated annealing that takes Fig. 1 as input. From (a) to (f) in Fig. 11, iteration counts are 2,500, 5,000, 75,000, 10,000, 12,500, and 15,000 respectively.

## 5.2 Interactive manipulation system

Our direct manipulation interface enables representation of all (single) parallel projections simply






input	module
 label overlaps other elements	label extraction
 solid lines and dotted lines are on a straight line	2D elements extraction
 0, X, Y or Z is used two or more times	coordinate axes identification
 3D position is not determined geometrically	3D reconstruction
 segments are used as modification	N/A

Fig. 10: Cases which need user correction

Table 1: Calculation time [s]

Input image	Fig. 1	Fig. 3
Label extraction	2.8	1.5
2D elements extraction	17.6	7.5
Axes identification	29.2	16.6
3D reconstruction	571.5	25.8

by dragging the tip position of the coordinate axes, as well as seamless and free crossing between parallel projections. For example, if the angles between two of three coordinate axes are all 120 [degree], then the result is an isometric projection. Furthermore, if the angles between  $x$  and  $z$  axes and  $z$  and  $y$  axes are 45 [degree] and the angles between  $y$  and  $x$  axes is 270 [degree], then the result is the cavalier projection.

Our tool can change viewpoint efficiently. For example, if a conventional 2D drawing tool is employed to change the second diagram into the third diagram in Fig. 1, the user must then move numerous line segments, change their forms, and move both points and labels. In contrast, if the user interface of this system is used, the same changes can be made simply by dragging the tip part of the  $x$  axis. Since the intermediate states are displayed while the tip of this coordinate axis is being dragged, the user can search continuously for the most suitable projection parameters. This technique rejuvenates the diagram manipulation process and makes possible large increases in efficiency.

## 6 Conclusion and Future Issues

In this paper, we proposed a novel technique of reconstructing 3D geometric diagrams from a single 2D image. In fields such as mathematics, engineering, and design, diagrams depicting 3D geometric information are frequently used as a means of communicating information that include geometrical

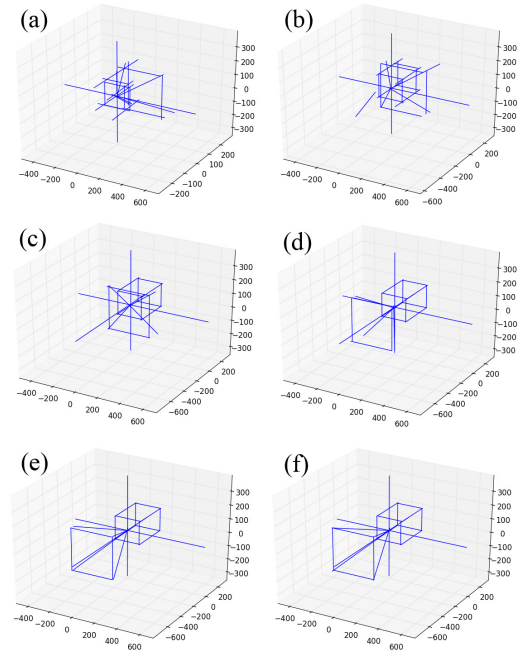


Fig. 11: Convergence of the simulated annealing.

concepts, composition and structure. Our system has broad applicability to platforms such as electronic publishing and Web applications.

While the current implementation applies to simple elements such as points and lines, we believe that the practical utility of this technique has been successfully demonstrated. There is no doubt, however, that its adaptation to elements such as curves and surfaces will contribute to further development of diagram representation techniques. For example, based on the assumption that each curve is planar, it should be possible to reconstruct curve elements by first reconstructing line segments connecting the two curve endpoints, and then projecting the curve to applicable plane [4]. Development of comprehensive techniques for curves, such as curve extraction, reconstruction and interfaces, will be the target of our future work.

Furthermore, if annotations can be added to the diagram manipulation interface, it may enable more intuitive manipulation. In the future, we will examine and analyze the potential for using annotations, including techniques for adding annotations and performing 3D modeling [35, 12]. Other possible subjects include developing techniques for automatically setting the optimal viewpoint for diagrams, and for creating new diagram representations by applying the interface proposed in this paper.

## References

- [1] K. Sugihara, *Machine Interpretation of Line Drawings*, MIT Press, 1986
- [2] M. Cooper, *Line Drawing Interpretation*, Springer, 2008
- [3] H. Lipson and M. Shpitalni, Optimization-based reconstruction of a 3D object from a single freehand line drawing, *Computer-Aided Design*, Vol. 28, No. 8, pp. 651-663, 1996
- [4] M. Masry, D. Kang and H. Lipson, A freehand sketching interface for progressive construction of 3D objects, *Computers and Graphics*, Vol. 29, No. 4, pp. 563-575, 2005
- [5] S. Lee, D. Feng and B. Gooch, Automatic Construction of 3D Models from Architectural Line Drawings, *Proc. I3D '08*, Vol. 123-130, 2008
- [6] J. Liu, L. Cao, Z. Li and X. Tang, Plane-Based Optimization for 3D Object Reconstruction from Single Line Drawings, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 30, No. 2, 315-327, 2008
- [7] T. Xue, J. Liu and X. Tang, Object cut: Complex 3D object reconstruction through line drawing separation, *Computer Vision and Pattern Recognition (CVPR)*, pp. 1149-1156, 2010
- [8] J. Liu, Y. Chen and X. Tang, Decomposition of Complex Line Drawings with Hidden Lines for 3D Planar-Faced Manifold Object Reconstruction, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 33, No. 1, pp. 3-15, 2011
- [9] L. Cao, J. Liu and X. Tang, What the Back of the Object Looks Like: 3D Reconstruction from Line Drawings without Hidden Lines, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 30, No. 3, pp. 507-517, 2008
- [10] R. C. Zeleznik, K. P. Herndon and J. F. Hughes, SKETCH: an interface for sketching 3D scenes, *SIGGRAPH '96*, pp. 163-170, 1996
- [11] T. Igarashi, S. Matsuoka and H. Tanaka, Teddy: a sketching interface for 3D freeform design, *SIGGRAPH '99*, pp. 409-416, 1999
- [12] M. Lau, G. Saul, J. Mitani and T. Igarashi, Modeling-in-context: user design of complementary objects with a single photo, *SBIM '10*, pp. 17-24, 2010
- [13] A. Rivers, F. Durand and T. Igarashi, 3D modeling with silhouettes, *ACM Trans. Graph.*, Vol. 29, No. 4, pp. 109:1-109:8, 2010
- [14] A. Criminisi, I. Reid and A. Zisserman, Single View Metrology, *International Journal of Computer Vision*, Vol. 40, No. 2, pp. 123-148, 2000
- [15] S. N. Sinha, D. Steedly, R. Szeliski, M. Agrawala and M. Pollefeys, Interactive 3D architectural modeling from unordered photo collections, *ACM Trans. Graph.*, Vol. 27, No. 5, pp. 159:1-159:10, 2008
- [16] D. Hoiem, A. A. Efros and M. Hebert, Automatic photo pop-up, *ACM Trans. Graph.*, Vol. 24, No. 3, pp. 577-584, 2005
- [17] N. Jiang, P. Tan and L.-F. Cheong, Symmetric architecture modeling with a single image, *ACM Trans. Graph.*, Vol. 28, No. 5, pp. 113:1-113:8, 2009
- [18] M. Agrawala, D. Zorin and T. Munzner, Artistic Multiprojection Rendering, *Eurographics Rendering Workshop 2000*, pp. 125-136, 2000
- [19] F. Grabler, M. Agrawala, R. W. Sumner and M. Pauly, Automatic Generation of Tourist Maps, *ACM Trans. Graph. (Proc. SIGGRAPH)*, Vol. 27, No. 3, pp. 100:1-100:11, 2008
- [20] P. Coleman and K. Singh, RYAN: Rendering Your Animation Nonlinearly projected, *NPAR '04*, pp. 129-156, 2004
- [21] D. N. Wood, A. Finkelstein, J. F. Hughes, C. E. Thayer and D. H. Salesin, Multiperspective panoramas for cel animation, *SIGGRAPH '97*, pp. 243-250, 1997
- [22] P. Rademacher and G. Bishop, Multiple-center-of-projection images, *SIGGRAPH '98*, pp. 199-206, 1998
- [23] V. Popescu, P. Rosen and N. A.-Villani, The graph camera, *ACM Transactions on Graphics (TOG)*, Vol. 28, No. 5, pp. 158:1-158:8 2009
- [24] J. Yu and L. McMillan, General Linear Cameras, *Computer Vision - ECCV 2004*, pp. 14-27, 2004
- [25] A. Adams and M. Levoy, General linear cameras with finite aperture, *EGSR'07*, pp. 121-126, 2007
- [26] R. Carroll, A. Agrawala and M. Agrawala, Image warps for artistic perspective manipulation, *ACM Trans. Graph.*, Vol. 29, No. 4, pp. 127:1-127:9, 2010
- [27] K. Singh, C. Grimm and N. Sudarsanam, The IBar: A Perspective-based Camera Widget, *Proc. UIST '04*, pp. 95-98, 2004
- [28] R. Schmidt, K. Singh and R. Balakrishnan, Sketching and Composing Widgets for 3D Manipulation, *cgforum*, Vol. 27, No. 2, pp. 301-310, 2008

- [29] Y. Zheng, H. Li and D. Doermann, A Parallel Line Detection Algorithm Based on HMM Decoding, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 27, No. 5, pp. 777-792, 2005
- [30] K. Tombre and S. Tabbone, Vectorization in graphics recognition: to thin or not to thin, 15th International Conference on Pattern Recognition, pp. 91-96, 2000
- [31] J. G. Daugman, Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters, *JOSA A*, Vol. 2, No. 7, pp. 1160-1169, 1985
- [32] OpenCV documentation, Image Filtering, <http://docs.opencv.org/trunk/modules/imgproc/doc/filtering.html>, Retrieved 2014-04-08
- [33] Z. Michalewicz and D. B. Fogel, *How to Solve It: Modern Heuristics*, Springer 2004
- [34] T. Segaran, C. Evans and J. Taylor, *Programming the Semantic Web*, O'Reilly Media Inc., 2009
- [35] Y. Gingold, T. Igarashi and D. Zorin, Structured annotations for 2D-to-3D modeling, *ACM Trans. Graph.*, Vol. 28, No. 5, pp. 148:1–148:9, 2009