Parameterized Sketches from Stereo Images

Danijela Markovic*, Efstathios Stavrakis and Margrit Gelautz

Interactive Media Systems Group, Institute for Software Technology and Interactive Systems
Vienna University of Technology, Favoritenstrasse 9-11/188/2, A-1040 Vienna, Austria

ABSTRACT

In this paper we present an algorithm to automatically generate sketches from stereo image pairs. Stereo analysis is initially performed on the input stereo pair to estimate a dense depth map. An Edge Combination image is computed by localising object contour edges, as indicated by the depth map, within the intensity reference image. We then approximate these pixel-represented contours by devising a new parametric curve fitting algorithm. The algorithm successfully recovers a minimum number of control points required to fit a Bézier curve onto the pixel-edge dataset in the Edge Combination image. Experiments demonstrate how the Edge Combination algorithm, used for dominant edge extraction, can be combined with a curve fitting algorithm to automatically provide parameterized artistic sketches or concept drawings of a real scene.

Keywords: Edges, object extraction, spline, sketching

1. INTRODUCTION AND MOTIVATION

Sketching has been an important medium of human expression and idea communication. Defined as a quick drawing that loosely captures the appearance of important scene features, it has been used as a study in preparation for larger, more detailed works of art. Sketching has also been the primary tool in the learning process of visual fine arts. Its importance, however, is evident in most activities of collaborative work where recording, development and sharing of ideas graphically can enhance human communication. More than a labor intensive drawing, sketching makes the information intimate and gives dominance of line over mass.

A lot of research has been devoted to simulation of human sketching ability to present various thoughts, ideas and to describe the world around just with few rough strokes. Several methods[1, 2, 3] have been used to analyze line drawings and to present them in a form more suitable for further manipulation. Also a great deal of research[4] is done to present existing artificial or real scenes in sketched form. Our focus in this paper is to present real scenes as stylized sketches by employing depth information to select only significant contours that play the most important role in object description. This dominant structure of an object appearance in a scene is represented further by using Bézier splines of a higher degree in order to have more freedom in depicting more complex shapes.

In this paper we present a method which can produce smooth line sketches from real recorded images. The input to the sketch algorithm is a natural scene, along with a user-defined set of parameters that define the quality of the image to be produced. The output is a smoothed line form of this scene with objects emphasized by their dominant edges. To extract only dominant edges in an image and to suppress unnecessary lines detected by edge detection, we use depth information as an additional source of information. Whereas a perfect depth map might be used to separate the object of interest from the background in a relatively straightforward way, in practical stereo analysis we have to cope with stereo matching errors that lead to erroneous depth values which are particularly present in the absence of texture and along object discontinuities.

In order to suppress the matching-induced errors along object boundaries, we suggest an Edge Combination algorithm that combines the edges derived from the disparity image with the original intensity edges for more accurate localization of contour edges.

Further author information: (Send correspondence to D.M.)
D.M.: markovic@tuwien.ac.at; Telephone +43 (1) 588 01 - 18859;
E.S.: stathis@tuwien.ac.at; Telephone +43 (1) 588 01 - 18872;
M.G.: gelautz@tuwien.ac.at; Telephone +43 (1) 588 01 - 18849;

2. ALGORITHM

A stereo image pair consisting of a left and right stereo image is processed by a stereo matching algorithm, which delivers as output a disparity map in the geometry of one of the two input images. For stereo matching, we use the technique[5], which matches individual scan lines of stereo pairs in epipolar geometry using dynamic programming. Next, an edge detector (e.g., Canny) is applied to the intensity image and disparity image, resulting in a set of intensity edges and disparity edges. The core of the processing pipeline is the Edge Combination algorithm that we devised in order to determine which edges in the intensity image are also present in the disparity edge image. The idea is to combine the edges from the stereo map, which suggest the location of scene object discontinuities, with the higher positional accuracy of the intensity edges for more accurate recognition of significant contour edges.

After the extraction of these combined edges, we approximate them by using Bézier curves. The approximation's function is twofold; the parametric representation of the contours allows flexible interactive manipulation of the extracted sketching lines, but also acts as a line stylization mechanism, since the data is fitted smoothly, and provides a distinct and artistic characteristic to the otherwise rigid edges. A summary of the involved processing steps can be seen in Fig. 1.

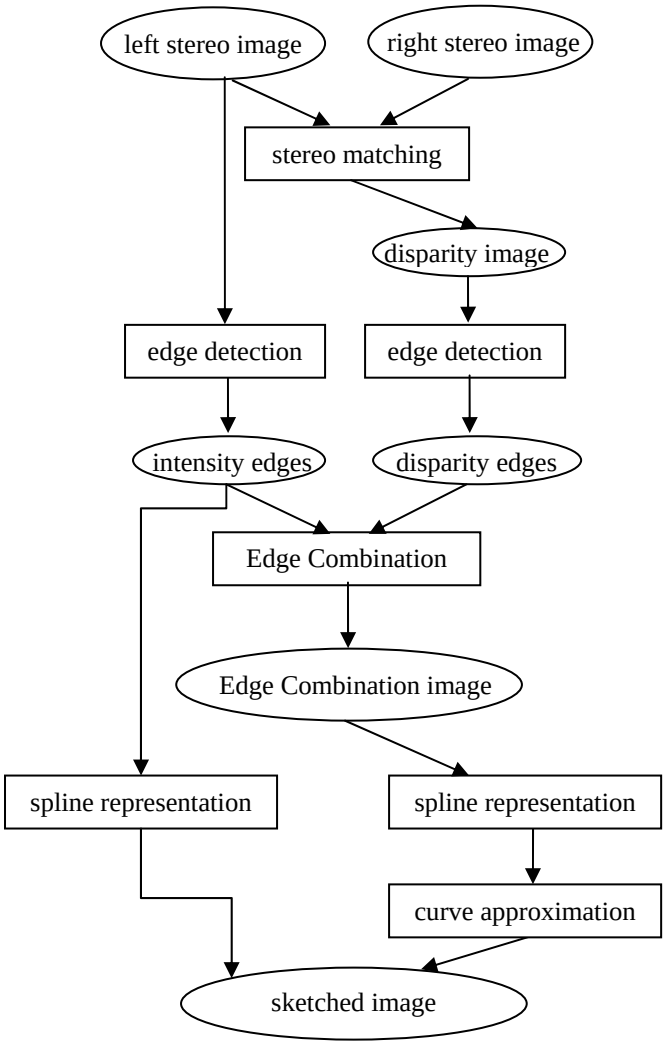


Figure 1. Overview of processing chain.

2.1 Edge Combination Algorithm

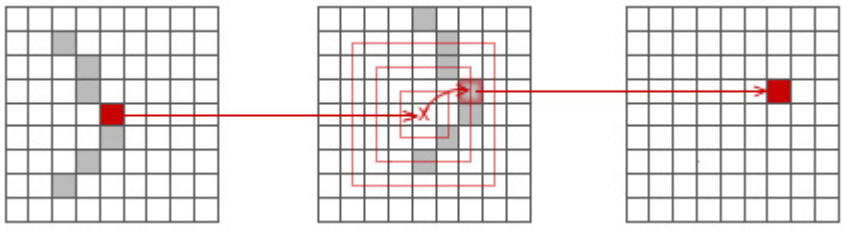
A pixel-by-pixel search is performed in the surrounding of each intensity edge to find a corresponding disparity edge. The search takes into account the orientation of the edge to reduce the possibility of mismatches. The result of this step is further refined during an edge linking process that bridges minor gaps in the computed combined edges based on a labeling of the connected edge components in the original intensity image. The linking procedure helps to suppress the effects of broken disparity edges caused by matching errors along object boundaries. The result of the Edge Combination step is a set of edges that are located along depth discontinuities, but with the positional accuracy of intensity edges. A detailed description of each distinct step of this procedure follows.

2.1.1 Edge detection

The first step is to detect edges in both the original image and its corresponding disparity image. We use the Canny edge detector, which we slightly modified to extract edges from color images. Apart from the pixel coordinates describing an edge pixel, we gather information about the orientation, in both the intensity and disparity edge images.

2.1.2 Edge search

The original edge image (I_E) and the disparity edge image (I_{DE}) are input to the Edge Combination procedure, shown in Fig. 2. For each edge pixel in I_{DE}, we determine whether a corresponding edge pixel with a similar orientation can be found in I_E. We use a search window of NxN, with $N=2k+1$ pixels, where $k \in [0,\ldots,k_{max}]$ and k_{max} is an integer parameter. Note that k should be progressively increased from 0 to a previously given maximum, k_{max}, until a matching pixel is found. To define similarity in edge orientation we usually employ a tolerance angle between 5° and 20°. We record every edge pixel in I_E that was found to have a corresponding edge pixel in I_{DE} in order to include it in the Edge Combination image. In this way, we build up a "basic" Edge Combination image I_{EC}.



(a) (b) (c)

Figure 2. Searching process in constructing the "basic" Edge Combination image: (a) I_{DE}, (b) I_E, (c) I_{EC}

2.1.3 Edge linking

Imperfect disparity information along continuous edges, usually sourcing from artifacts in the stereo matching process, will prevent some pixels in the comparing process to match, leaving a gap in the reconstructed contour line. Gaps of this type are corrected with an edge linking procedure which repairs broken edges in I_{EC}, when a continuous edge in I_E indicates that edge segments should be connected.

First, we use a labeling algorithm to determine the connected edge components in I_E. For each end point of an edge in I_{EC}, we search within a neighborhood of MxM, with $M=2k_L+1$ pixels, where $k_L \in [0,\ldots,k_{Lmax}]$, to find another end pixel in I_{EC}. If both of them belong to the same edge in I_E, as determined by the previous labeling, we connect the two end points in I_{EC} by inserting the corresponding edge segment from I_E. In practice, we copy an appropriate subwindow of size $M_{max}xM_{max}$, with $M_{max}=2k_{Lmax}+1$ pixels from I_E and insert it into I_{EC}. Before insertion, we clean the subwindow by pruning superfluous parts of the copied edge pattern using the cleaning technique described in the following. The edge linking procedure terminates, if no more open end points that could be connected can be found in I_{EC}.

2.1.4 Cleaning (Maze solving technique)

To remove unneeded line parts from the subwindow we identified as the one to be copied in the previous step, Fig. 4(a), we use the Maze solving technique[6]. For every end pixel of the subwindow as shown in 3(b) we check if it is one of the end pixels that we initially wanted to connect, marked in 3(a). If this test fails we remove that pixel from the subwindow and continue testing the remaining end pixels, as shown stepwise in 3(b) through 3(h). This procedure terminates when we only find end pixels that have the same position as the pixels that we want to connect, marked in Fig. 3(h).

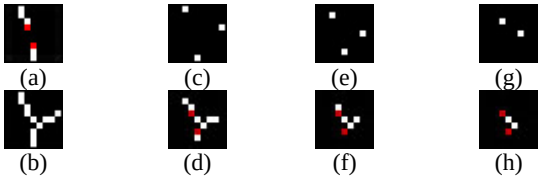


Figure 3. Subwindow cleaning.

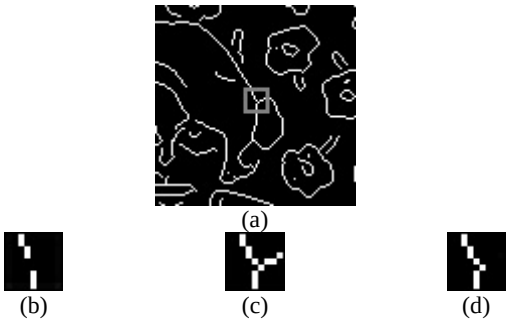


Figure 4. (a) Part of the original edge image, (b) Edge Combination image before linking, (c) Edge Combination image after linking and before cleaning, (d) Edge Combination image after linking and cleaning.

2.2 Converting Rasterized Lines to Splines

The edges in the final Edge Combination image I_{EC} are properly localized around significant edges. However, a bitmap representation of these lines is usually not sufficiently flexible for stylization. Therefore we convert the rasterized contours to a parametric representation by fitting Bézier curves. As a starting point to our line-to-spline conversion algorithm we use ideas from Bhuiyan and Hama[7]. Our algorithm, however, strives for economy in data representation by selecting the degree of the Bézier curves that best suits the approximation of each edge line, as well as performance by using adaptive parameters. The concept of our algorithm is to initiate a Bézier curve and by adjusting its shape we minimize the distance between the original edge and the approximating curve. To adjust the shape we move the control points and calculate the new error, accepting iteratively the new location of the control points if they generate a curve where the estimated error is smaller than the previous one.

2.2.1 Initialization

For positioning the initial control points of the approximating spline (*A*), we use a smooth spline representation (*B*) of the edges by considering all pixels of a line as spline control points, with *a* and *b* points of the sets *A* and *B* respectively. This representation is used to determine the direction and curvature[8] at each point, as presented in Fig. 5. The curvature is calculated by the angle between two elementary segments: $\beta(n) = \alpha(n+1) - \alpha(n-1)$ which we then use for extreme points' (P_k) detection.

We place an initial control point P'_k at a distance of *4H*, in the direction of the height vector *H* of a triangle formed by points P_{k-1}, P_k and P_{k+1}, as shown in Fig. 6. From our experiments we determined that a distance of *4H* is a good starting position to decrease the computation of the searching process.

Figure 5. (a) Direction, (b) curvature estimation.

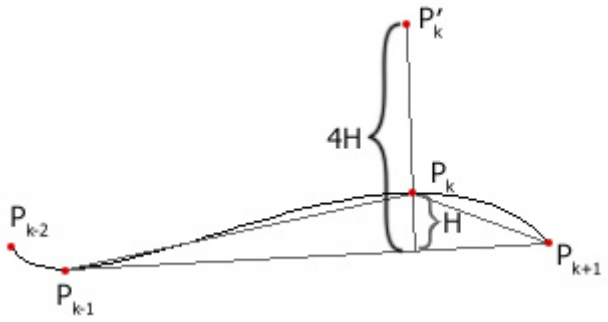


Figure 6. Control points initialization.

2.2.2 Curve approximation

To approximate the edge curve with the initialized Bézier spline, we move all the intermediate control points to new locations until a better approximation of the original edge dataset is achieved. For each intermediate control point $P'_k(x,y)$, we translate it to four possible new locations:

(a) $P'_k(x+h,y)$,

(b) $P'_k(x-h,y)$,

(c) $P'_k(x, y+h)$,

(d) $P'_k(x, y-h)$,

where h is an adaptive moving distance parameter associated with an error metric. This metric is the largest distance between the two curves calculated by using the Hausdorff distance[9], given as:

$$h(A, B) = \max_{a \in A} \min_{b \in B} d(a, b) , \quad (1)$$

where $d(a,b)$ we define as the maximum of the heights of the triangles constructed by each point on set A and its two closest points on B.

Whenever the error is decreased by a movement we accept the new control point location and reiterate until a maximum number of user-defined iterations has been reached, or the error has become sufficiently small, under a user-defined threshold.

2.2.3 Line stylization

We have so far extracted a set of dominant edges from the original scene and converted them into an efficient parametric form. We now stylize this parameterized edge representation by imitating the stroke pressure at each curve point. We apply a stroke pressure equal to the maximum estimated error, that is the Hausdorff distance as described in the previous section, at that point of the approximated curve for which we estimated this error, see Fig.7(a). We then interpolate this pressure along the stroke path, setting the pressure to zero at the starting and ending point of the curve. This procedure produces a stroke tapering effect, as can be seen in Fig.7(b).

(a) (b)

Figure 7. (a) Final maximum error measured by the Hausdorff distance between the approximated spline and original curve, (b) final style of the line.

3. TESTS AND RESULTS

3.1. Test Data

For our tests, we employed a stereo configuration of two Dragonfly IEEE-1394 video cameras as delivered by Point-grey[10]. The frames are provided to the user in 24 bit RGB format. The camera set-up was calibrated using the calibration routines provided by Intel's Open Source Computer Vision Library[11]. We then transformed the image pairs into epipolar geometry to facilitate the subsequent stereo analysis.

3.2. Experimental Results

The application of the method to our test data and the results obtained are illustrated in Figs. 8 through 10. Figures 8 (a) and (b) show a pair of stereo video frames in epipolar geometry. The size of the images is 400 x 400 pixels.

For our tests, we utilized an implementation of the Pixel-to-Pixel stereo matching algorithm described by Birchfield and Tomasi[5]. The resulting disparity map is given in Fig. 8 (c). Figs. 8 (d) and 8(f) show the edges derived from the original intensity image 8(a) and the disparity map 8(c), respectively. Fig. 8(e) contains the contour edges computed by the Edge Combination approach. One can recognize the smoother appearance of the combined edges in 8(e) when compared to the stereo edges in 8(f). Fig. 8(g) shows a final sketch result with the background of the image produced by representing edge lines of the real image by Bézier curves and foreground strokes of dominant edges of the scene computed by the Edge Combination algorithm, represented by the previously explained approximation procedure and stylized to imitate artistic stroke pressure discussed in Section 2.2.3. Final result illustrates fast look of a real scene by using the dominance of significant lines artistically highlighted as a smooth hand-drawn effect.

4. CONCLUSIONS

In this paper we presented a method to utilize depth-from-stereo information to extract object contours from real images. In this context, we implemented a fast curve approximation technique to convert these rasterized edges into a parametric form. The parametric edges, outlining meaningful objects in the scene are then stylized by using strokes to produce sketches that have a hand-crafted appearance. The smoothness of the stylized output is similar to that found in many concept drawings and sketches. The algorithm can assist an artist in creating the basis for a fine art work or it can help a beginner artist as a guiding tool on important features of a scene when learning visual fine arts.

ACKNOWLEDGEMENTS

Major parts of this work were supported by Hochschuljubiläumsstiftung of the City of Vienna under the project "Image and Video Processing for Artistic Effects" (H-1153/2003). Co-author Efstathios Stavrakis was funded by the Austrian Science Fund (FWF) under project P15663.

REFERENCES

1. T. Sezgin, T. Stahovich, and R. Davis, "Sketch Based Interfaces: Early Processing for Sketch Understanding", in Proceedings of the *Workshop on Perceptive User Interfaces*, Orlando, Florida, 2001.
2. M. Shpitalni and H. Lipson, "Classification of Sketch Strokes and Corner Detection Using Conic Sections and Adaptive Clustering", *Trans. of ASME J. of Mechanical Design*, vol. **119**(2), pp. 131-135, 1996.
3. Bo Yu and Shijie Cai, "A Domain-Independent System for Sketch Recognition", in Proceedings of the *1st International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, Melbourne, Australia, pp. 141 - 146, 2003.
4. O. Tolba, J. Dorsey, and L. McMillan, "Sketching with Projective 2d Strokes", in *Proceedings of UIST 99*, ACM SIGCHI, Asheville, North Carolina, United States, vol. **1**(1), pp. 149-157, 1999.
5. S. Birchfield and C. Tomasi, "Depth Discontinuities by Pixel-to-Pixel Stereo", *International Journal of Computer Vision*, vol. **35**(3), pp. 269-293, 1999.
6. B. Nayfeh, "Cellular Automata for Solving Mazes", *Doctor Dobb's Journal*, vol. **18**(2), pp. 32-38, 1993.
7. A.-A. Bhuiyan and H. Hama, "Recovering the Control Points of Bézier Curves for Line Image Indexing", *Journal of Electronic Imaging*, vol. **11**(2), pp. 177-186, 2002.
8. R. Gross, "Run-On Recognition in an On-Line Handwriting Recognition System", Term Project, University of Karlsruhe, 1997.

9. G. Rote, “Computing the Minimum Hausdorff Distance Between Two Point Sets on a Line Under Translation”, *Information Processing Letters*, vol. **38**(3), pp. 123-127, 1991.
10. Point Grey Research Inc., http://www.ptgrey.com, 2004.
11. Intel Open Source Computer Vision Library, v. 3.1, http://www.intel.com/research/mrl/research/opencv/, 2004.

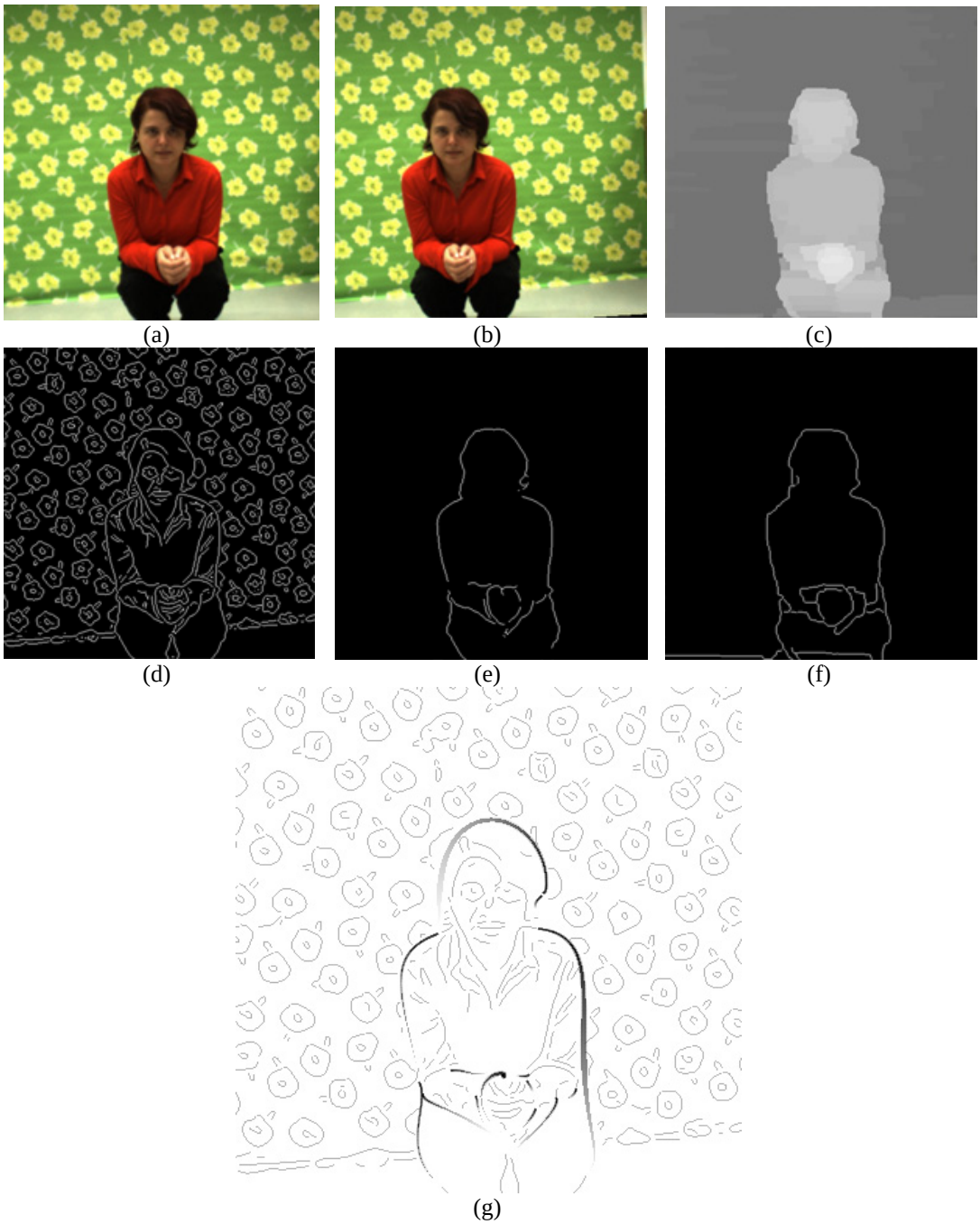


Figure 8. (a) Left camera image, (b) right camera image, (c) disparity image, (d) original edge image, (e) Edge Combination image, (f) disparity edge image, (g) sketched image.

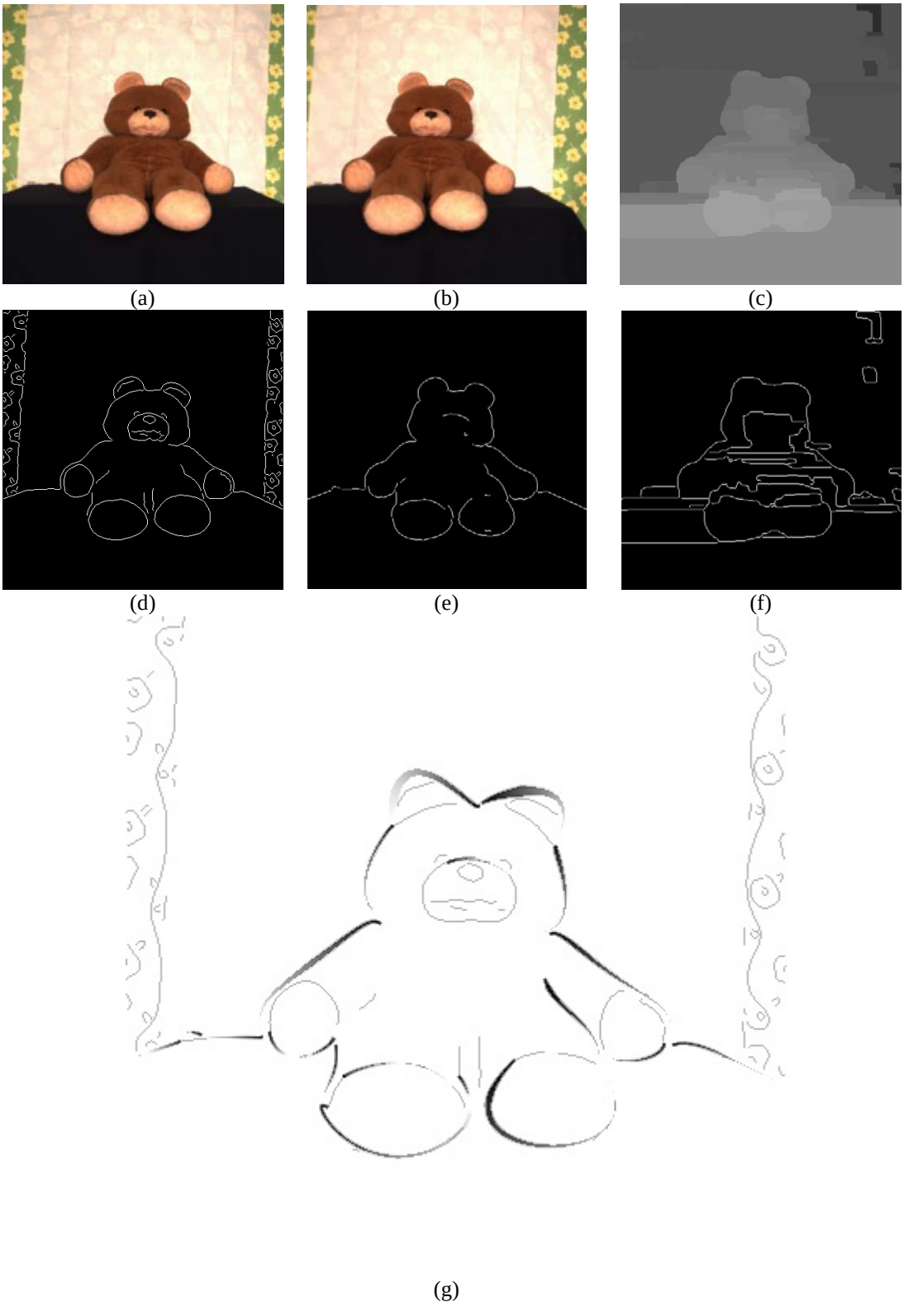


Figure 9. (a) Left camera image, (b) right camera image, (c) disparity image, (d) original edge image, (e) Edge Combination image, (f) disparity edge image, (g) sketched image.

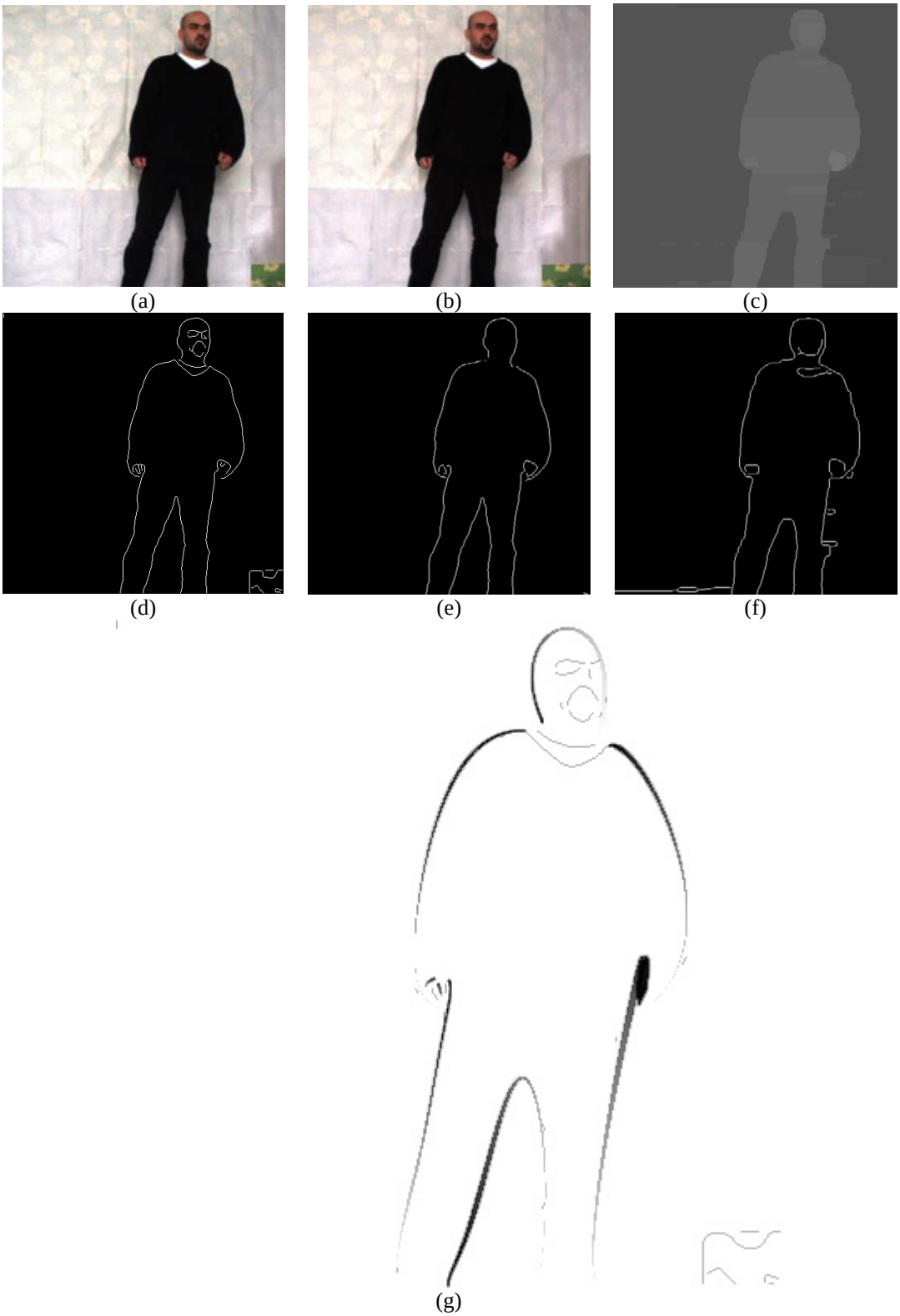


Figure 10. (a) Left camera image, (b) right camera image, (c) disparity image, (d) original edge image, (e) Edge Combination image, (f) disparity edge image, (g) sketched image.