# Image segmentation to inspect 3-D object sizes

**Jui-Pin Hsu**
**Chiou-Shann Fuh**
National Taiwan University
Department of Computer Science and Information Engineering
Taipei, Taiwan
E-mail: fuh@robot.csie.ntu.edu.tw

**Abstract.** Object size inspection is an important task and has various applications in computer vision. For example, the automatic control of stone-breaking machines, which perform better if the sizes of the stones to be broken can be predicted. An algorithm is proposed for image segmentation in size inspection for almost round stones with high or low texture. Although our experiments are focused on stones, the algorithm can be applied to other 3-D objects. We use one fixed camera and four light sources at four different positions one at a time, to take four images. Then we compute the image differences and binarize them to extract edges. We explain, step by step, the photographing, the edge extraction, the noise removal, and the edge gap filling. Experimental results are presented.

*Subject terms: visual communications and image processing; image segmentation; image difference; edge extraction; edge detection; gap filling.*

*Optical Engineering 35(1), 262–271 (January 1996).*

## 1 Introduction

### 1.1 *Motivation*

A typical stone-breaking machine has two rollers to squeeze and break stones, as shown in Fig. 1. To use the machine, we should adjust the space between the two rollers properly, according to the sizes of the stones. On one hand, if large stones come, we should adjust the space wider, or the lifetime of the rollers will be shortened; on the other hand, if small stones come, we should adjust the space narrower, or the stones will slip through, and electricity will be wasted. Hence, segmentation on stone images for size inspection is important if we want automatic control of the machine.

### 1.2 *Summary*

If the stones have some high-contrast texture, a traditional gradient edge detector will find many false edges. If the stones have almost no texture, it will be hard for a stereo image technique to match corresponding points.[1] Similarly, a traditional gradient edge detector will fail to find the edges between two stones with almost the same brightness. Thus we use four light sources to the north, south, west, and east of the camera, one at a time, to photograph images and then use image differences to detect the shadows, which occur at the edges, for edge extraction. The image differences are computed for each pair of images with light sources at opposite sides, to extract edges in both north-south (N-S) and east-west (E-W) directions.

Some related works on edge detection are Refs. 2, 3. They extract edges from a source image rather than from image differences.

Treating image differences as edge responses, we binarize them to get binary edge images in both N-S and E-W directions. Usually, a single-threshold scheme cannot acquire good edge images. We propose an original binarization scheme using local and global thresholds.

After the binarization comes the edge imaging, implemented as the binary OR of binary edge images in the N-S and E-W directions. After the edges are extracted, we remove the noise by connected-component analysis, to get a clearer edge image.

Due to some inevitable problems explained later, the extracted edges will not all be linked. General edge-linking techniques[4] seem unable to solve the problem. A snake algorithm[5] or front propagation algorithm[6] may fill the gaps but will fall into local minima, as shown in Fig. 2. We detect the terminals of edge pieces by a corner detector using a $K$-cosine algorithm,[7,8] and we propose a terminal extension algorithm to fill the edge gaps. Some related proposals for corner detection are in Refs. 9 to 12. They may be chosen to improve performance in certain cases of segmentation, but we choose the original $K$-cosine algorithm for simplicity, and it suffices.

## 2 Setup for Photographing

We are to take four images with four light sources to the north, south, east, and west of the camera, one at a time. Figure 3 shows the setup with the light source to the east ($E$) of the camera. The other light source positions are similar, and to the north ($N$), south ($S$), and west ($W$), respectively. The four source images are shown in Fig. 4, which was taken with a CCD camera with lens of focal length 25 mm. The resolution is 479H$\times$632W, and the distance between lens and the stones is about 62 cm. Each light source is about 7 to 8 cm away from the center of the lens. In our experiments, the sizes of the stones are about $5\times5\times2$ to $8\times6\times5$ cm. The light source is a frosted 40-W light bulb, which provides

Paper VIS-05 received May 28, 1995; revised manuscript received Aug. 18, 1995; accepted for publication Aug. 23, 1995.

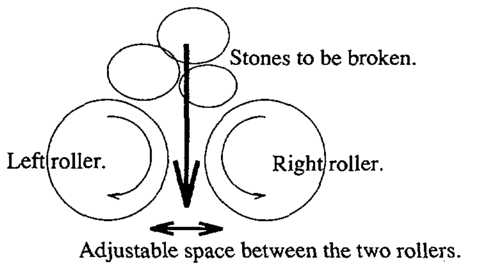


**Fig. 1** Side view of the rollers of a stone-breaking machine.

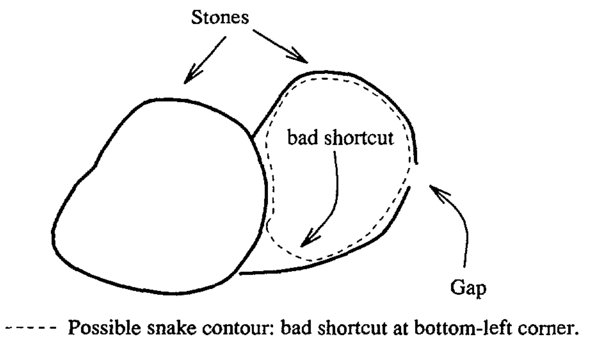

Stones
bad shortcut
Gap
----- Possible snake contour: bad shortcut at bottom-left corner.

**Fig. 2** Snake algorithm will fall into local minima in some cases.

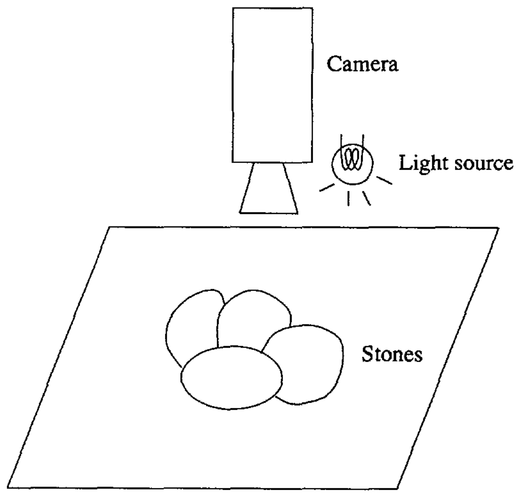


**Fig. 3** Global view of the camera setup.

equal illuminance in all directions. Light bulbs of clear glass will generate uncontrollable patterned light and degrade the subsequent processing of image differences.

## 3 Algorithm and Experimental Results

### 3.1 *Absolute Image Difference*

Let the four source images be $N$, $S$, $E$, and $W$, corresponding to the images with light sources to the north, south, east, and west of the camera, respectively. We can see that the major differences between the four images are in the positions of the shadows. We compute the absolute image differences, $\|N-S\|$ and $\|E-W\|$, where $\|\cdot\|$ stands for absolute value, to extract edges from the shadows.

As we can expect, the absolute image difference $\|N-S\|$ will have larger values at the horizontal edges of the stones, but smaller values at edges in other directions and non-edge pixels, since the shadows will differ at horizontal edges. Similarly, $\|E-W\|$ has the same effect at vertical edges. Figure 5 shows the histogram equalized absolute image differences for viewing.

### 3.2 *Local and Global Thresholding*

To get binary edge images in both N-S and E-W directions, we binarize the absolute image differences. Using a single global threshold will result in edge images with large gaps between broken edge segments. Thus we combine local thresholding and global thresholding to extract edges and get edge images with small gaps.

For local thresholding, we define a local thresholding condition, to test if a pixel has its intensity $\alpha\%$ higher than the average intensity of its local $13\times13$ window area or not. Those that pass the local thresholding condition will be marked as white; the others as black. Applying local thresholding to $\|N-S\|$ and $\|E-W\|$, we get two images, $\mathrm{LTHR}_{\mathrm{ns}}$ and $\mathrm{LTHR}_{\mathrm{ew}}$, as shown in Fig. 6. The percentage $\alpha$ is set to 20 in our experiments.

For global thresholding, we binarize images, with threshold at a little higher percentage ($\beta\%$) than the expected percentage ($\gamma\%$) of edge pixels. The percentage $\gamma$ is typically about 15, and $\beta$ is set to 22 in our experiment. That is, the brightest 22% of pixels will pass the global threshold condition and be marked as white, and the others as black. Applying global thresholding to $\|N-S\|$ and $\|E-W\|$, we get two images, $\mathrm{GTHR}_{\mathrm{ns}}$ and $\mathrm{GTHR}_{\mathrm{ew}}$, as shown in Fig. 7.

To combine local thresholding and global thresholding, we compute $\mathrm{THR}_{\mathrm{ns}}$ and $\mathrm{THR}_{\mathrm{ew}}$, as shown in Fig. 8, by the following equations:

$$\mathrm{THR}_{\mathrm{ns}}=\mathrm{LTHR}_{\mathrm{ns}}\ \text{AND}\ \mathrm{GTHR}_{\mathrm{ns}}\ ,$$

$$\mathrm{THR}_{\mathrm{ew}}=\mathrm{LTHR}_{\mathrm{ew}}\ \text{AND}\ \mathrm{GTHR}_{\mathrm{ew}}\ .$$

Here AND denotes the binary AND operator.

To gather edge information on both the N-S and E-W directions, we compute the combined edge image, $\mathrm{THR}_{\mathrm{combined}}$, as shown in Fig. 9, by the following equations:

$$\mathrm{THR}_{\mathrm{combined}}=\mathrm{THR}_{\mathrm{ns}}\ \text{OR}\ \mathrm{THR}_{\mathrm{ew}}\ .$$

Here OR denotes the binary OR operator.

We do not set the global threshold value at the exact percentage of edge pixels, not only because it is hard to know the exact percentage, but also because we rely highly on local thresholding, which extracts edges more effectively. Global thresholding only helps us to eliminate the false edges extracted by local thresholding at pixels of small intensities. Thus a somewhat loose condition for global thresholding will allow more edge pixels, and also noise pixels, to pass the threshold, and leave them to be distinguished by local thresholding. Local thresholding is unstable at pixels of small intensities, since when the intensity is small, a little change in intensity will make a large change in percentage, and our local thresholding is based on relative intensity. In the local-thresholding results (Fig. 6), we can see many false edge

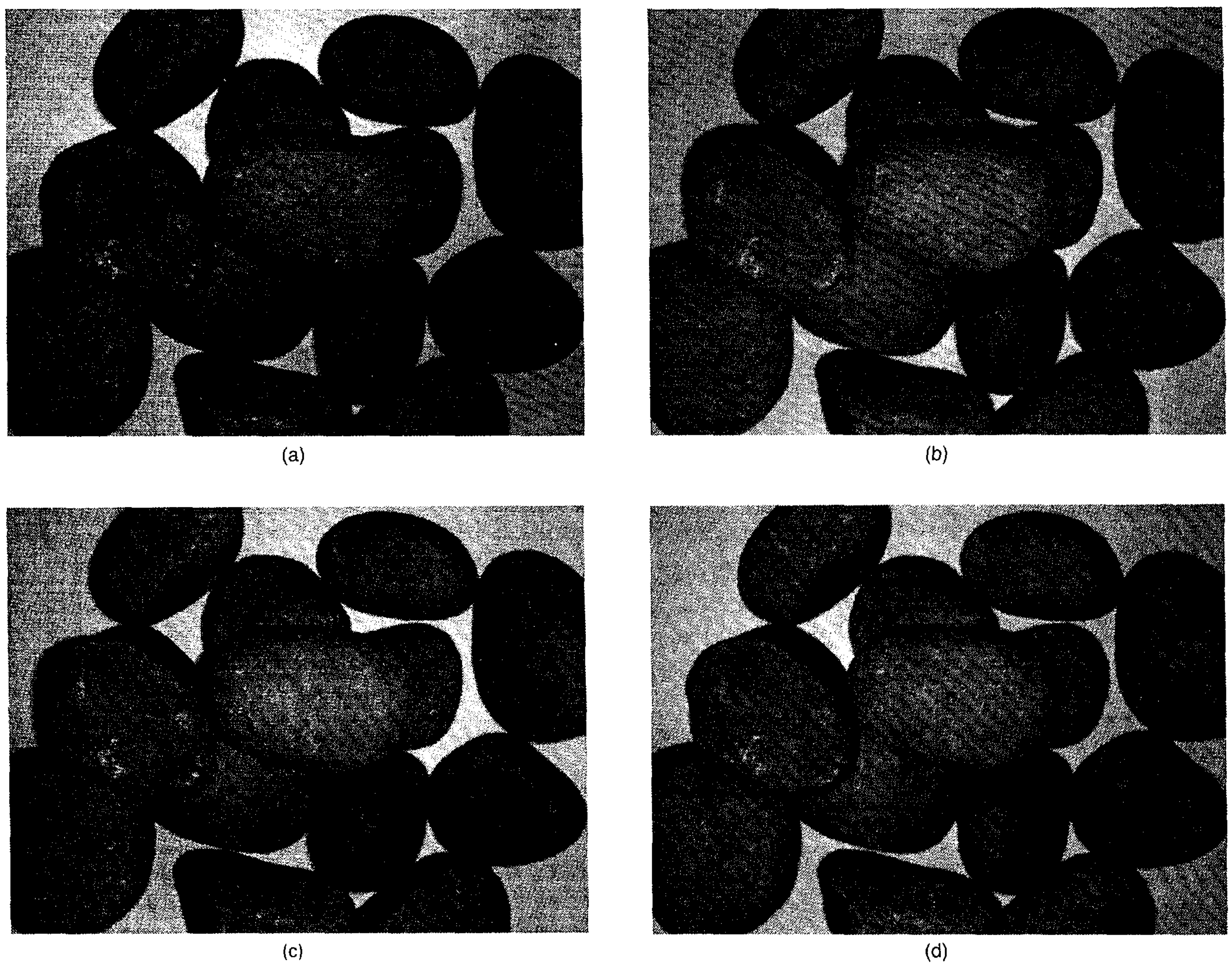


**Fig. 4** Source images with light sources to four directions (a) N, (b) S, (c) E, and (d) W to the camera. The resolution is 479H×632W.

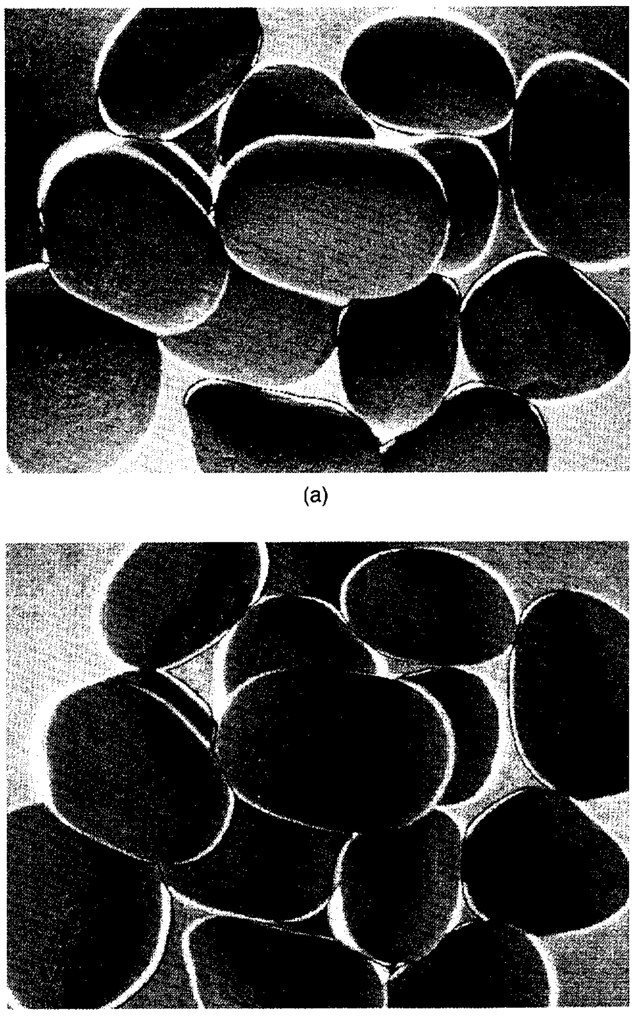
pixels at the locations with low intensities in image differences (Fig. 5). However, the false edge pixels cannot pass the global threshold (Fig. 7) in either the N-S or the E-W direction.

Stones generally are very Lambertian, but not perfectly Lambertian. Illuminating an object that is not Lambertian will result in bright spots in the image, caused by direct reflection of the light source. The positions of the bright spots change as the positions of the light sources change. Thus we will get significant image differences at the bright spot areas, and obtain false edges, if only a single global threshold is adopted. Because stones are very Lambertian, the intensity of pixels in bright spot areas will change smoothly. Thus pixels in bright spot areas cannot pass the local thresholding condition, and will be marked as non-edge pixels. We can see many false edge pixels in global thresholding results (Fig. 7) in bright spot areas of the source images (Fig. 4). However, the false edge pixels cannot pass local threshold (Fig. 6) in either the N-S or the E-W direction.

For choosing the parameter $\alpha$, the stronger the contrast between shadows and stones in the source image is, the better the edge image will be, and the higher the value at which $\alpha$ can be set. Specifically, what we want to distinguish in the thresholding is the edge pixels and the non-edge pixels. In the absolute image differences, edge pixels come from differences in brightness between shadows and stones, while non-edge pixels come from differences in brightness between stones illuminated by light sources at different positions. Thus the parameter $\alpha$ depends on the contrast of the two differences, while the two differences themselves depend only on the illumination, including the gray levels of the stones and the brightness of the light source. Thus the darker the stones are, the smaller the difference of the two differences will be, the worse the edge image will be, and the lower the value at which $\alpha$ can be set.

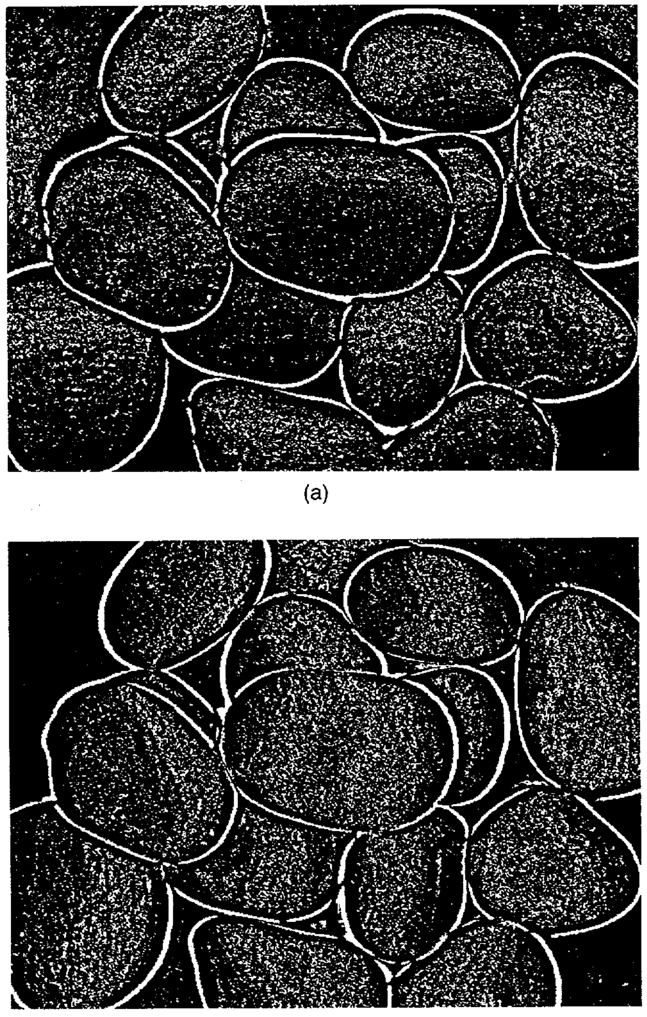
The parameter $\beta$ can be set to a percentage a little higher than the expected percentage of edge pixels, and $\beta$ has little effect, because global thresholding only helps local thresholding at the pixels of low intensity in image difference.

The window size for local thresholding depends on the thickness of the shadows, which depends on the thickness of the stones and the distance between the light source and the


(a)

(b)

**Fig. 5** Histogram-equalized absolute image differences for viewing: (a) Hist ($\|N-S\|$). (b) Hist($\|E-W\|$). Hist ($\cdot$) stands for histogram equalization.


(a)

(b)

**Fig. 6** Local thresholding: Pixels 20% brighter than average brightness in the local $13\times13$ area will pass the threshold and be marked as white. (a) $\mathrm{LTHR}_{ns}=\mathrm{LThr}_{20\%}(|N-S|)$, and (b) $\mathrm{LTHR}_{ew}=\mathrm{LThr}_{20\%}(|E-W|)$, where $\mathrm{LThr}_{20\%}$ ($\cdot$) stands for our local thresholding with $\alpha=20$.

stones. The thicker the stones are and the shorter the distance between the light source and stones, the thicker the shadows will be, and the larger the window size should be set. In our experimental results, a window size of $7\times7$ is good, and can filter out much of the noise. For demonstrating possible noise spots, and some typical bad cases, we use a window size of $13\times13$.

## 3.3 *Noise Removal for the Edge Image*

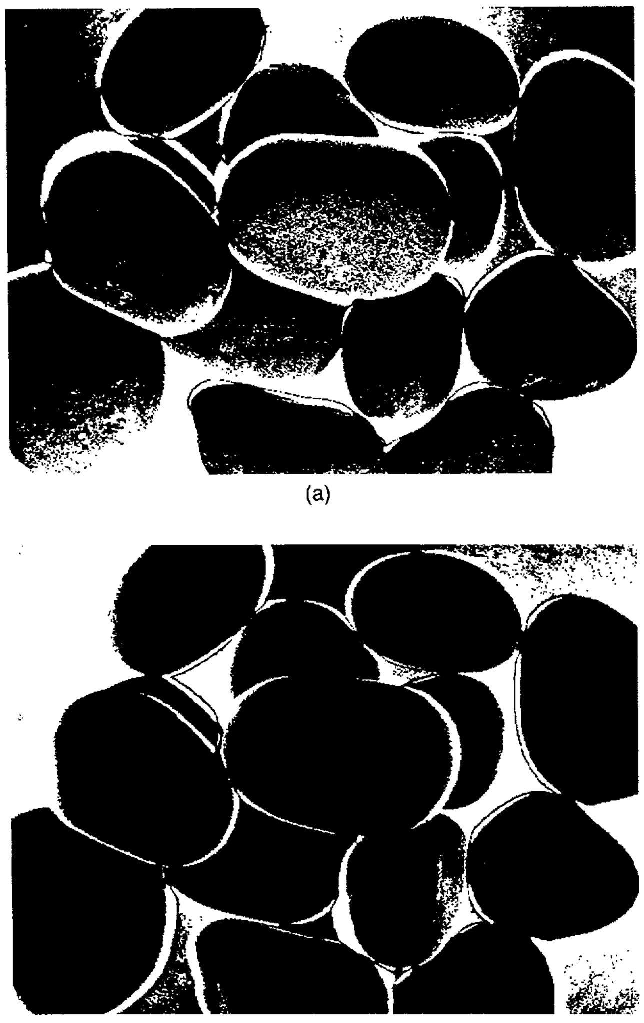
After the thresholding, the edge image has small spots of noise. For noise removal, we use the connected component algorithm on $\mathrm{THR}_{\mathrm{combined}}$ to filter out small white components less than 40 pixels and small black components less than 400 pixels. The result image, EDGE, is shown in Fig. 10. We assumed that stones whose areas are smaller than 400 pixels do not exist. In practice, stones of such small sizes are unimportant and thus ignorable. Some noise components exceeding 40 pixels may exist. This problem can be solved by a convex hull algorithm after the segmentation. We discuss it in Section 4.2.

## 3.4 *Smoothing the Edge Boundaries and Using a K-Cosine Algorithm to Extract Edge Terminals*

The edge image has many gaps. Figure 11 shows one of the reasons why edge gaps form. We trace the edge boundaries and find the edge terminals to fill the edge gaps.

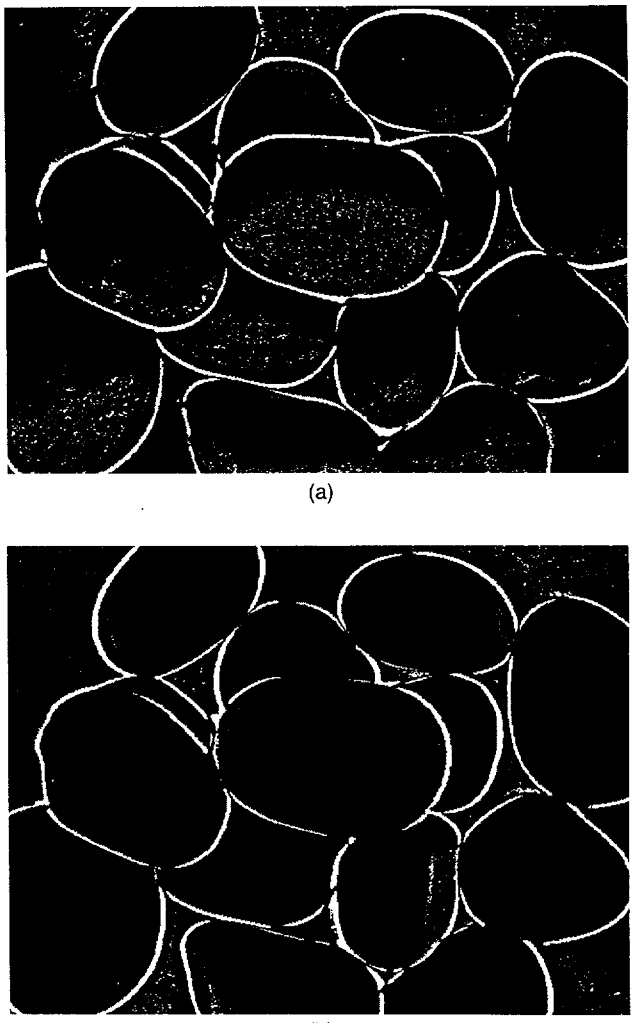
For smoothness, we dilate each edge pixel in EDGE to all its eight neighbors, called $\mathrm{EDGE}_{\mathrm{thick}}$, as shown in Fig. 12. This results in a little less accurate edge image; however, it provides more accurate $K$-cosine values for later processing.

Figure 13 shows the boundaries of edges. For each pixel along the boundaries, we compute the $K$-cosine values by tracing the boundary counterclockwise. Each edge pixel is classified as a left turn or right turn, according to the angle used to compute the $K$-cosine value. By thresholding $K$-cosine values we can find corners, at which we select the right-turn pieces to be the terminal pieces of the edges, as shown in Fig. 14.


(a)

(b)

**Fig. 7** Global thresholding: The brightest 22% of pixels will be marked white. (a) $GTHR_{ns}=GThr_{22\%}(|N-S|)$, and (b) $GTHR_{ew}=GThr_{22\%}(|E-W|)$, where $GThr_{22\%}(\cdot)$ stands for our global thresholding with $\beta=22$.


(a)

(b)

**Fig. 8** Combination of local thresholding and global thresholding: Pixels passing both local and global thresholds will be marked white and otherwise black. (a) $THR_{ns}$, binary AND of Fig. 6(a) and Fig. 7(a), (b) $THR_{ew}$, binary AND of Fig. 6(b) and Fig. 7(b).

## 3.5 *Terminal Extension to Fill Gaps*

Let the two end points of the terminal pieces be $P_1$ and $P_2$, the midpoint of the piece be $P_{\text{mid}}$, the midpoint of $P_1$ and $P_2$ be $P_{\text{mean}}$, and the vector **V** be $\overrightarrow{P_{\text{mean}}P_{\text{mid}}}$ Let the two unit vectors **U** and **W** have angle $\phi$ between each of them and **V**. We search the triangle spanned by $s$**U** and $s$**W**, where $s$ is a scalar, to fill the gaps, as explained in Fig. 15. If a white pixel that is not locally connected to $P_{\text{mid}}$ is found in the triangle, we decide the space between the white pixel and $P_{\text{mid}}$ is the gap, and directly link the white pixel to $P_{\text{mid}}$ by a straight line segment. Contrarily, if no white pixels are found in the triangle, we draw a white line segment from $P_{\text{mid}}$ to the midpoint of the other side of the triangle, that is, the altitude of the triangle from $P_{\text{mid}}$. The result is shown in Fig. 16. The parameter $\phi$ is set to 35 deg, and $s$ is set so that the altitude of the triangle described above is 30, that is, $s=30$ $\sec\phi=30\sec(35\text{ deg})\approx 36.6$. Note that if a gap of length 50 is encountered, and the gap has two terminal pieces detected

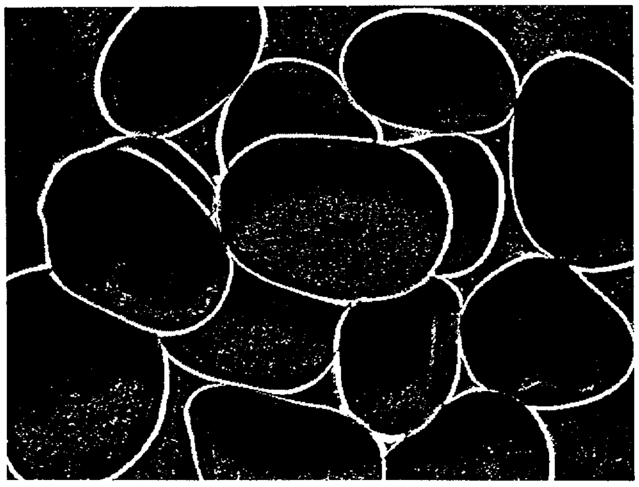


**Fig. 9** $THR_{\text{combined}}$, gathering N-S and E-W information: binary OR of Figs. 8(a) and 8(b).

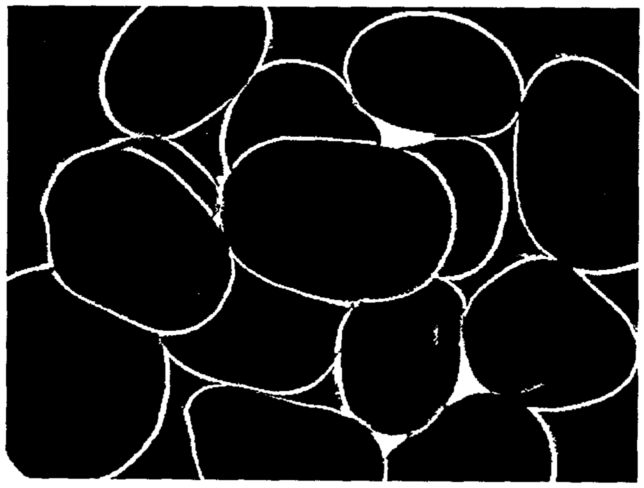


**Fig. 10** EDGE, noise-removed edge image: White connected components less than 40 pixels and black connected components less than 400 pixels in $THR_{combined}$ are filtered out.

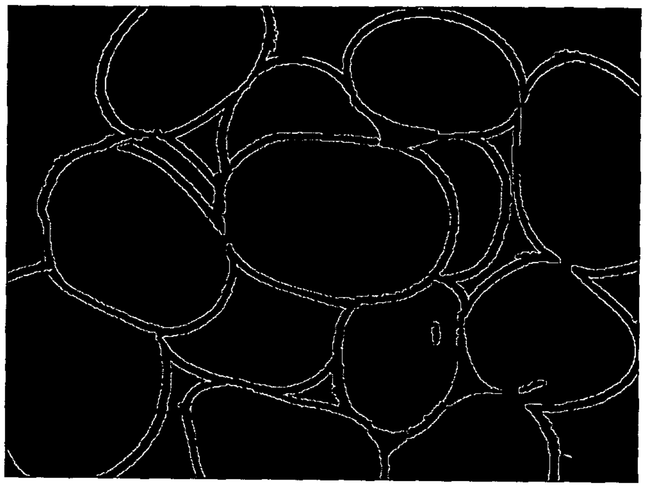


**Fig. 13** Boundary of the dilated edge images. $EDGE_{boundary}$, boundaries of $EDGE_{thick}$: to be traced.

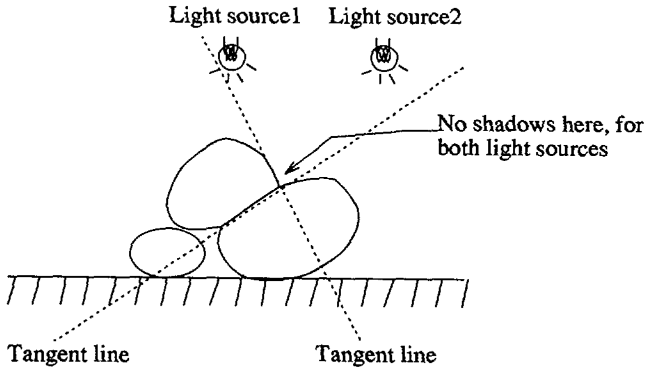


**Fig. 11** No shadow exists under each of the two light sources at the same sides of both tangent lines, so no significant difference is on the pointed edge. Thus some edge pixels will not be detected, and edge gaps form.

**Fig. 14** Terminal pieces of edge pieces. White pixels: detected terminals of edge pieces; black pixels: nonterminals of edge pieces; and gray areas: nonedge pixels.

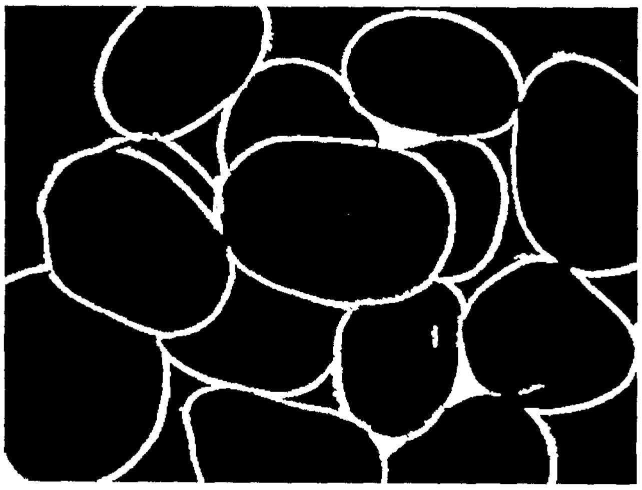


**Fig. 12** Dilation of edge image to smooth edge boundaries.

by our algorithm, then the terminal extension algorithm will extend the first terminal piece by 30, since the search for edge pixels fails. On the second terminal piece, the algorithm will search for and find the extension part of the first terminal, and then link them by a line segment, since the gap will be reduced to about 20 after the first extension. Thus the two line segments fill the gaps.

We can expect that the best chosen $\phi$ is independent of stone size, and a little dependent on gap size. In experiments, we find that $\phi$ is not sensitive to noise, stone size, or gap size, and can be set to a constant. The parameter $s$ depends on gap size. The larger the gaps are, the higher the value at which $s$ should be set.

## 3.6 *Removal of Unwanted Filling Line Segments*

Some line segments for gap filling may be false. The unwanted line segments are all thin, and the pixels on the thin line will have less white pixels in their neighborhood than

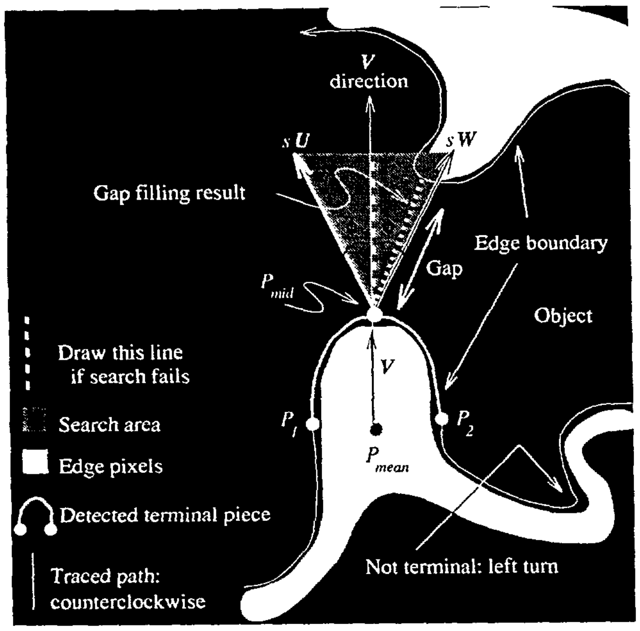


**Fig. 15** Terminal extension algorithm. See text.

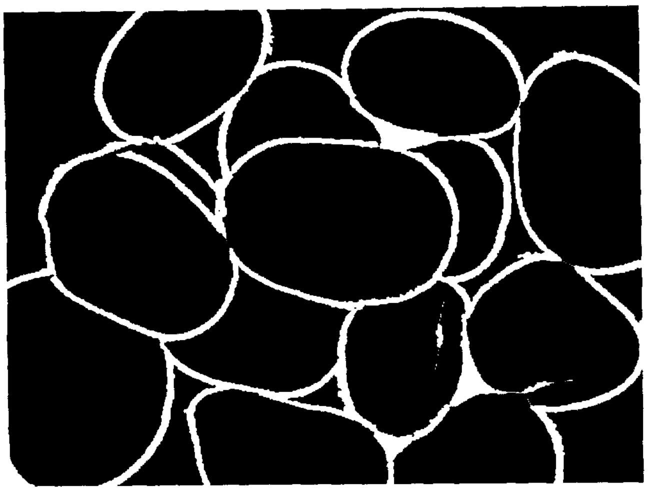


**Fig. 16** Result of terminal extension for gap filling. The line segments fill the gaps.

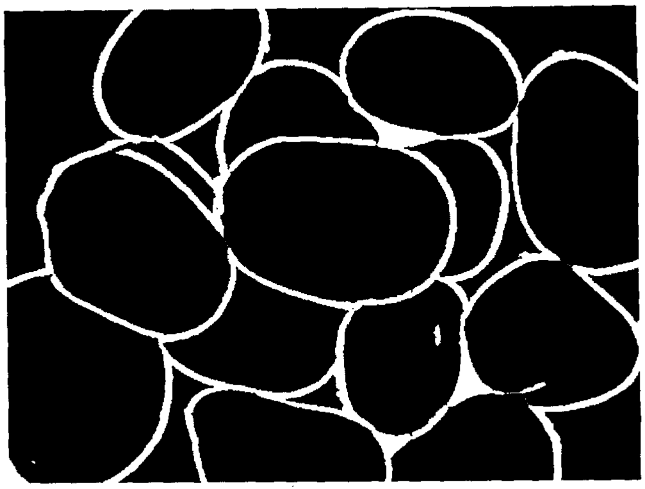


**Fig. 17** The complete segmentation result: unwanted line segments removed from the gap filling result.

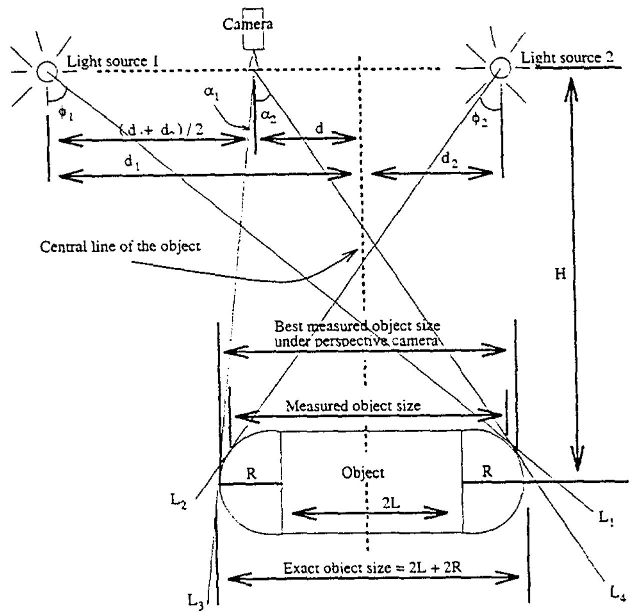


**Fig. 18** Geometrical and radiometric model.

normal thick boundary pixels, as illustrated in Fig. 16. Thus we can detect the pixels on the thin lines, and eliminate the ones that are not separators of any two or more regions. Figure 17 shows the result, and we complete the whole segmentation.

## 3.7 *Biased Estimation*

It should be noted that this method is a biased estimation, as shown in Fig. 18. Ordinarily, stones will not be perfectly round, so our object model consists of two arcs and two straight lines. The object model will shrink to a ball when $L$ equals zero. The part of the object to the right of the tangent line $L_1$ will be in shadow under light source 1 but will be bright under light source 2. This part will be detected as edges. Similarly, the part to the left of $L_2$ will be detected as edges. Thus the size of the part of the object between the two tangent lines $L_1$ and $L_2$ will be the measured object size (MOS). Since only the part of object between the tangent lines $L_3$ and $L_4$ can be viewed by a perspective camera, the size of this part will be the best measurable object size (BMOS). The exact object size (EOS) is $2L+2R$. The relations are as follows:

$$[(d_1+L)^2+H^2-R^2]^{1/2} \sin\phi_1 = R \cos\phi_1 + L + d_1 \ , \qquad (1)$$

$$[(d_2+L)^2+H^2-R^2]^{1/2} \sin\phi_2 = R \cos\phi_2 + L + d_2 \ , \qquad (2)$$

$$\mathrm{MOS} = R \cos\phi_1 + R \cos\phi_2 + 2L \ , \qquad (3)$$

$$[(d-L)^2+H^2-R^2]^{1/2} \sin\alpha_1 = R \cos\alpha_1 + L - d \ , \qquad (4)$$

$$[(d+L)^2+H^2-R^2]^{1/2}\sin\alpha_2=R\cos\alpha_2+L+d\ , \tag{5}$$

$$\mathrm{BMOS}=R\cos\alpha_1+R\cos\alpha_2+2L\ , \tag{6}$$

$$d=\frac{d_1-d_2}{2}\ . \tag{7}$$

In some cases other than that in the figure, when the related positions of the camera, the light, and the object differ, the formulas may differ in certain signs (plus or minus) for some certain terms. The relations are a little complex. Some typical parameters for the bias are listed in Table 1.

### 3.8 *Performance*

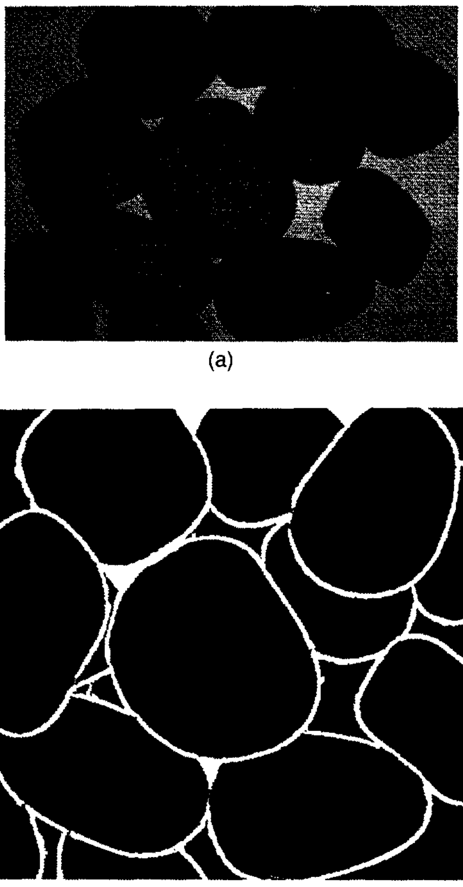
The algorithm above is implemented on a Sun SPARCstation 10 machine. Although the program is not optimized, it can still complete the work in 50 s. In fact, it should be done in 7 s, including the computation of the stone size, which is not included in this paper. The time can be shortened still further by better hardware, an optimized program, or perhaps a parallel program on a parallel computer. More experimental results are shown in Figs. 19 to 23.

## 4 Conclusion and Future Work

### 4.1 *Conclusion*

We have proposed an algorithm, including picture photographing, edge extraction, noise removal, and edge gap filling, for stone image segmentation. Our key idea is to use image differences to be able to process typical stones with both high and low texture. For edge gap filling, we make use of a *K*-cosine algorithm to find edge terminal pieces.

Black stones whose brightness is very similar to that of the shadows cannot be processed successfully by this algorithm, since we use the image difference between brightness of stones and shadows to detect edges. Similarly, it is also hard to detect edges of black stones with human eyes. The stones to be broken are usually brown or gray in the real world. Another application is to break stone-shaped concrete pieces, which are not black either.


(a)

(b)

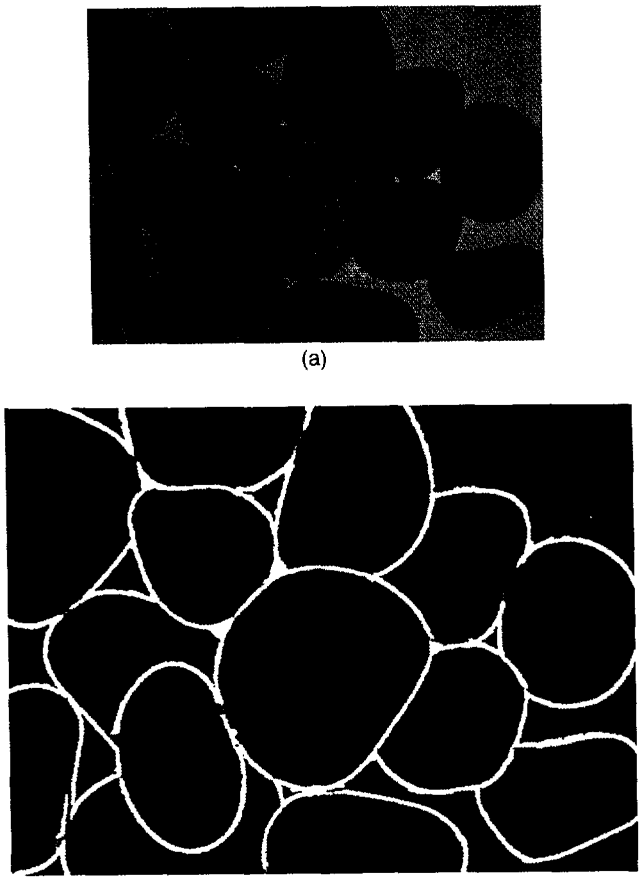
**Fig. 19** Experiment 1: (a) one of the four source images; (b) the segmentation result.

### 4.2 *Future Work*

In the thresholding of absolute image differences, the parameters α and β are set manually. A prospect for future work is automatically choosing the value of α; the value of β affects

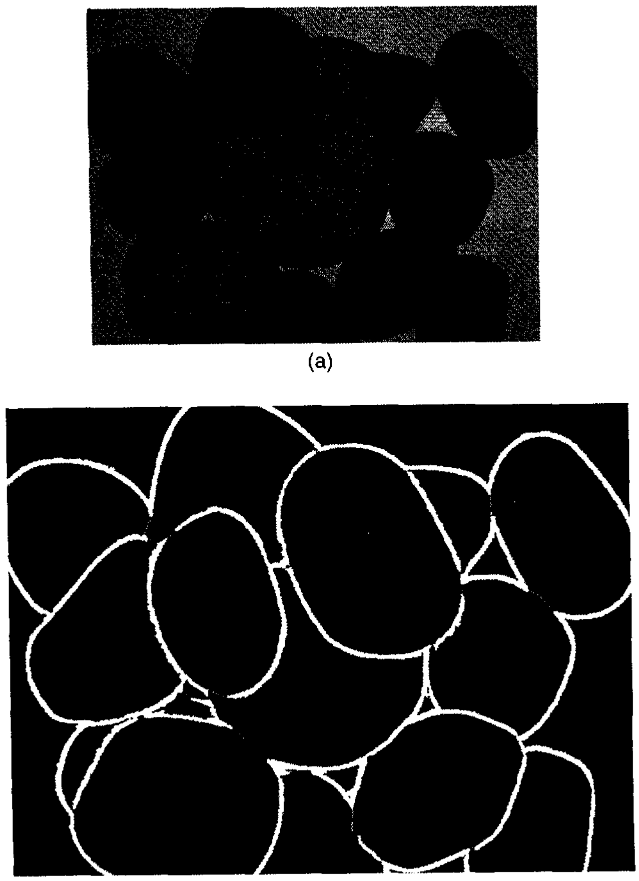
**Table 1** Typical parameters versus error rates: the B rate stands for (MOS – BMOS)/BMOS, and the E rate stands for (MOS – EMOS)/EMOS.

| Free Variable | | | | | Constrained Variable | | | | | | | | Error Rate | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $H$ | $L$ | $R$ | $d_1$ | $d_2$ | $d$ | $\phi_1$ | $\phi_2$ | $\alpha_1$ | $\alpha_2$ | $MOS$ | $BMOS$ | $EOS$ | B Rate | E Rate |
| 62 | 1.5 | 1 | 8 | 8 | 0 | 0.168 | 0.168 | 0.040 | 0.040 | 4.972 | 4.998 | 5.000 | -0.5% | -0.5% |
| 62 | 1.5 | 1 | 9 | 7 | 1 | 0.184 | 0.152 | 0.024 | 0.056 | 4.971 | 4.998 | 5.000 | -0.5% | -0.6% |
| 62 | 0.5 | 2.5 | 8 | 8 | 0 | 0.151 | 0.151 | 0.024 | 0.024 | 5.941 | 5.999 | 6.000 | -0.9% | -1.0% |
| 62 | 0.5 | 2.5 | 10 | 6 | 2 | 0.182 | 0.119 | -0.008 | 0.056 | 5.941 | 5.996 | 6.000 | -0.9% | -1.0% |
| 62 | 0.5 | 2.5 | 18 | -2 | 10 | 0.301 | -0.008 | -0.137 | 0.182 | 5.887 | 5.935 | 6.000 | -0.8% | -1.9% |


(a)

(b)

**Fig. 20** Experiment 2: (a) one of the four source images; (b) the segmentation result.


(a)

(b)

**Fig. 21** Experiment 3: (a) one of the four source images; (b) the segmentation result.

the results only slightly, and is less critical. For estimating α automatically, the discussion of the principles for its choice in Section 3.2 may be helpful.

To estimate sizes of stones, which is one direction for future work, we should first estimate the missing edges and occluding edges. Although the line segments for gap filling cannot represent the true edges of the stones, we can retrace the edges of the stones after the segmentation, and use the resulting edge information, except the line segments, to estimate the missing edges, including the improper line segments and the occluded edges. The analysis of these edges will help to eliminate the small noise spots of size exceeding 40 pixels mentioned in Sec. 3.3. After all this, sizes of the stones can easily be estimated.

## Acknowledgments

This research work was supported by National Science Council of Taiwan, ROC, under NSC grants NSC 83-0422-E-002-010 and NSC 84-2212-E-002-046, and by Cho-Chang Tsung Foundation of Education under grant 84-S-26.

## References

1. R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision,* Vol. 2, pp. 357–362, Addison-Wesley, Reading, MA, 1993.
2. K. R. Rao and J. Ben-Arie, "Optimal edge detection using expansion matching and restoration," *IEEE Trans. Pattern Anal. and Machine Intell.* **16**(12), 1169–1182 (Dec. 1994).
3. F. van der Heijden, "Edge and line feature extraction based on covariance models," *IEEE Trans. Pattern Anal. and Machine Intell.* **17**(1), 16–33 (Jan. 1995).
4. R. C. Gonzalez and R. E. Woods, *Digital Image Processing,* pp. 429–443, Addison-Wesley, Reading, MA, 1992.
5. F. Leymarie and M. D. Levine, "Simulating the grassfire transform using an active contour model," *IEEE Trans. Pattern Anal. and Machine Intell.* **14**(1), pp. 56–75 (Jan. 1992).
6. R. Malladi, J. A. Sethian, and B. C. Vemuri, "Shape modeling with front propagation: a level set approach," *IEEE Trans. Pattern Anal. and Machine Intell.* **17**(2), 158–175 (Feb. 1995).
7. A. Rosenfeld and E. Johnston, "Angle detection on digital curves," *IEEE Trans. Comput.* **C-22,** 875–878 (Sep. 1973).
8. A. Rosenfeld and J. S. Weszka, "An improved method of angle detection on digital curves," *IEEE Trans. Comput.* **24,** 940–941 (1975).
9. X. Li and N. S. Hall, "Corner detection and shape classification of on-line hand-printed kanji strokes," *Pattern Recognition* **26**(9), 1315–1334 (1993).
10. W. Y. Wu and M. J. Wang, "Detecting the dominant points by the curvature-based polygonal approximation," *CVGIP: Graphical Models and Image Process.* **55**(2), 79–88 (1993).
11. A. Rattarangsi and R. T. Chin, "Scale-based detection of corners of planar curves," *IEEE Trans. Pattern Anal. and Machine Intell.* **14**(4), 430–449 (Apr. 1992).
12. S. J. Wang and C. S. Fuh, "Adaptive dominant point detection via the rated composite vectors," Master's Thesis, Dept. of Computer Science and Information Engineering, National Taiwan Univ., Taipei, Taiwan (June 1994).

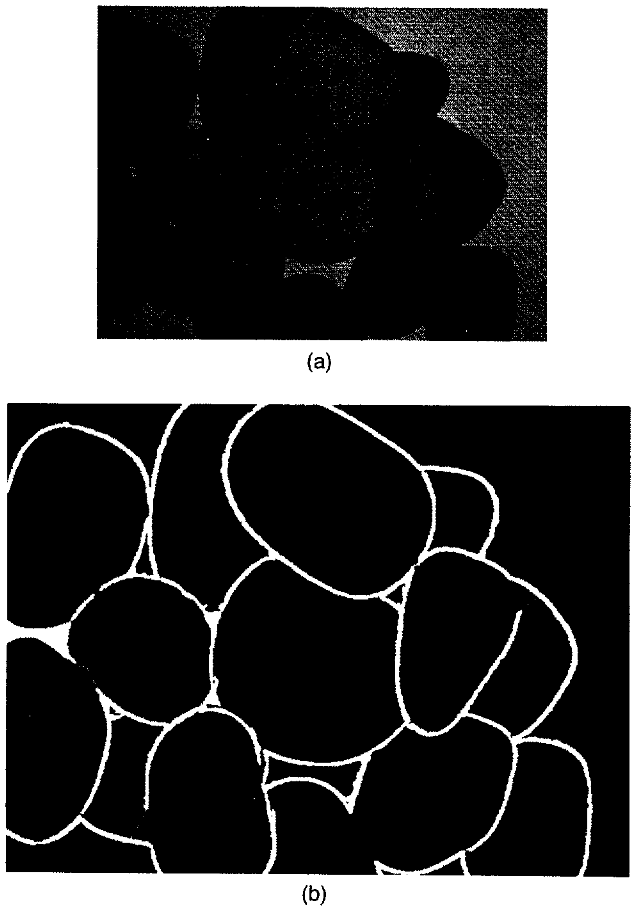


**Fig. 22** Experiment 4: (a) one of the four source images; (b) the segmentation result.

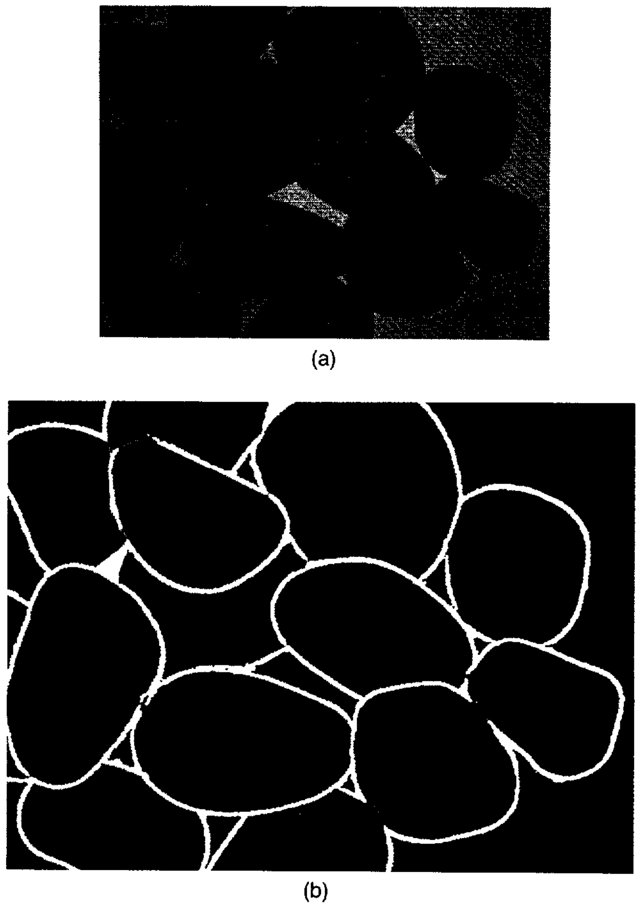
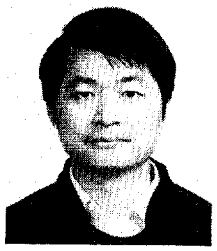


**Fig. 23** Experiment 5: (a) one of the four source images; (b) the segmentation result.

**Jui-Pin Hsu** received the BS degree in mathematics from National Taiwan University, Taipei, Taiwan, in 1992, and the MS degree in computer science and information engineering from the same university in 1995. He is now a lecturer in Foo-Yin Junior College of Nursing and Medical Technology, Kaohsiung, Taiwan. His current research interests include image segmentation, image reconstruction, and computer vision.



**Chiou-Shann Fuh** received the BS degree in computer science and information engineering from National Taiwan University, Taipei, Taiwan, in 1983, the MS degree in computer science from the Pennsylvania State University, University Park, in 1987, and the PhD degree in computer science from Harvard University, Cambridge, MA, in 1992. He was with AT&T Bell Laboratories and engaged in performance monitoring of switching networks from 1992 to 1993. Since 1993, he has been an associate professor in the Computer Science and Information Engineering Department at National Taiwan University, Taipei, Taiwan. His current research interests include digital image processing, computer vision, pattern recognition, and mathematical morphology.