Hindawi Publishing Corporation
Journal of Sensors
Volume 2016, Article ID 5328130, 10 pages
http://dx.doi.org/10.1155/2016/5328130

Research Article

Edge Detection from RGB-D Image Based on Structured Forests

Heng Zhang, Zhenqiang Wen, Yanli Liu, and Gang Xu

School of Information Engineering, East China Jiaotong University, Nanchang 330013, China

Correspondence should be addressed to Yanli Liu; hbliuyanli@126.com

Received 6 January 2016; Revised 23 April 2016; Accepted 8 June 2016

Academic Editor: Virender K. Sharma

This paper looks into the fundamental problem in computer vision: edge detection. We propose a new edge detector using structured random forests as the classifier, which can make full use of RGB-D image information from Kinect. Before classification, the adaptive bilateral filter is used for the denoising processing of the depth image. As data sources, information of 13 channels from RGB-D image is computed. In order to train the random forest classifier, the approximation measurement of the information gain is used. All the structured labels at a given node are mapped to a discrete set of labels using the Principal Component Analysis (PCA) method. NYUD2 dataset is used to train our structured random forests. The random forest algorithm is used to classify the RGB-D image information for extracting the edge of the image. In addition to the proposed methodology, the quantitative comparisons of different algorithms are presented. The results of the experiments demonstrate the significant improvements of our algorithm over the state of the art.

1. Introduction

Edge is identified as an abrupt change in some low-level image feature such as color or intensity. Edge detection belongs to classification problem [1]. According to the feature, an image pixel can be classified as an edge pixel or not. So the core of the problem is to design a good classifier. There are three methods based on the image types: gray image edge detection algorithm, color image edge detection algorithm, and RGB-D image edge detection algorithm.

The edge of the gray image is reflected by the change of the edge gray value. This variation is generally reflected in the roof change or step change. In the mathematics, it is reflected in first derivative and second derivative. Therefore, there are two main types of edge detection algorithm for gray images: first-order differential image edge detection operator, such as Sobel operator [2], Prewitt operator [3], Kirsch operator [4], and Roberts operator [5], and second-order differential edge detection operator, like Laplacian operator, LOG operator [6], and others such as Canny operator [7] and SUSAN operator [8].

Compared to the gray images, color images contain more RGB information and luminance information. The edge of the color image is a collection of pixels that image color changes dramatically in the local area. There are two methods of edge detection: scalar operation and vector operation. Scalar operation converts the RGB vector of each pixel to scalar processing. In scalar operation, one method is to convert color image into gray image; another is three-channel method based on gray method, which divides the color image into three channels. Each channel is calculated by gray level method and the edge of color image is synthesized by three channels according to predefined rules. In vector operation, the RGB values of each pixel in the color image are considered as the vector integral. In [9], the gray operator Prewitt is extended to vector space. The gradient calculation method of gray image is introduced to the color image by Di Zenzo [10]. They calculate partial derivative to get the gradient amplitude and direction of the color image. The paper [11] extends quaternion method to the color image edge detection method.

With the development of imaging device, the image acquisition technology is getting better and better. Depth image acquisition is becoming cheaper and more popular. The algorithm of edge detection based on RGB-D image is becoming more mature. In the paper [12], on the basis of the gPb-ucm (Globalized Probability of Boundary-Ultrametric Contour Map) algorithm [13], the algorithm combines the depth information, RGB information, and texture information for each direction. The support vector machines (SVMs)

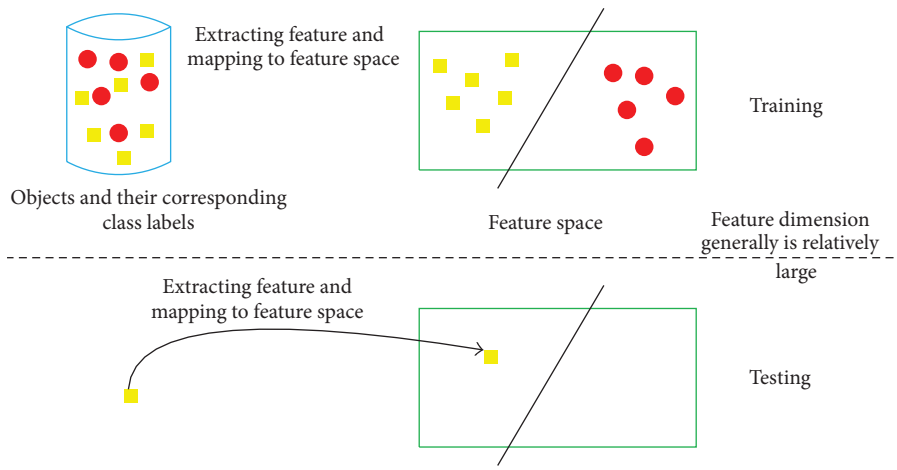

Figure 1: Process of edge detection.

with additive kernels [14] are used as the classifier. Dollar and Zitnick [15] consider the depth information and the RGB information (but not the texture information) and select the structured random forest as the classifier, which is used for edge detection. Its emphasis is on structural representation of the random forest rather than the representation of feature information. The depth information is used as a separate image for the classification integration with the geometrical features. In the paper [16], proposed approach firstly preprocesses the original depth map of objects in indoor environments captured by an RGB-D camera system and then uses the recovered depth map to detect sharp corners/edges on objects and calculate their sharpness. The paper [17] proposes a method that extracts occlusion edge in RGB-D frames by deep Convolutional Neural Networks. It avoids hand-crafting of features for occlusion edges detection. The paper [18] proposes a new probabilistic model, Contour Completion Random Fields, which allows completing the boundaries of occluded surfaces.

In this paper, we propose an improved RGB-D image edge detection algorithm based on structured forests [12, 15]. We use adaptive bilateral filter to denoise the depth image and make full use of the RGB-D image information and solve the problem that the outline of the former is not obvious [12] and the defect that the algorithm [15] misses details of the edge.

The rest of the paper is organized as follows: Section 2 refers to the information representation. Section 3 introduces in detail structured forest algorithm. Section 4 describes edge detection from RGB-D image. Section 5 shows the experimental results. Section 6 sets out the conclusions and presents lines for future work.

2. Representation

The Kinect sensor incorporates several advanced sensing hardware items. Most notably, it contains a depth sensor, a color camera, and a four-microphone array that provides full-body 3D motion capture, facial recognition, and voice recognition capabilities.

Figure 1 introduces the process of edge detection. According to Figure 1, our work is divided into two parts. Firstly, we study how to describe the color information and the depth information of the image. Secondly, we learn how to classify these features' information and how to extract the edge of the image.

For each pixel, color features are studied, which include brightness gradient (BG), color gradient (CG), and texture gradient (TG). We also estimate its 3D location in the scene and its surface normal orientation. The local geometric information is used to calculate the edge information of each pixel in three directions. A depth gradient (DG) indicates a discontinuity of depth information, a convex normal gradient (NG+) identifies the surface convex at a specified point in a specified direction, and a concave normal gradient (NG−) identifies the surface concaves at a specified point in a specified direction.

Regarding color gradient and texture gradient in gPb (Globalized Probability of Boundary) [12], they are important to RGB-D image. These data mainly have the following three rules: (1) it is a nonlinear noise model as $|\delta_Z| \propto Z^2|\delta_d|$, where δ_Z is the measurement error of depth information, Z is the actual depth information, and δ_d is the noncontinuity error of depth observation (depending on the triangulation principle of Kinect). This model leads to the systematic quantization and the nonrandomness of the depth information. (2) The lack of time synchronization between color channel and depth channel leads to the deviation of dataset. (3) There is lack of the depth information observations. We design the geometric edge information extraction scheme with the physical interpretation in detail. We use the multiscale window analysis, namely, adaptive joint bilateral filter, rather than using interpolation to make up missing depth information. In this system, least square method is used to fit the disparity gradient instead of the points in the point cloud, and Savitzky and Golay [19] parabolic is used to fit independent smoothing orientation.

In order to estimate the local geometric edge information, a disk is centered for each image. The disk is divided into

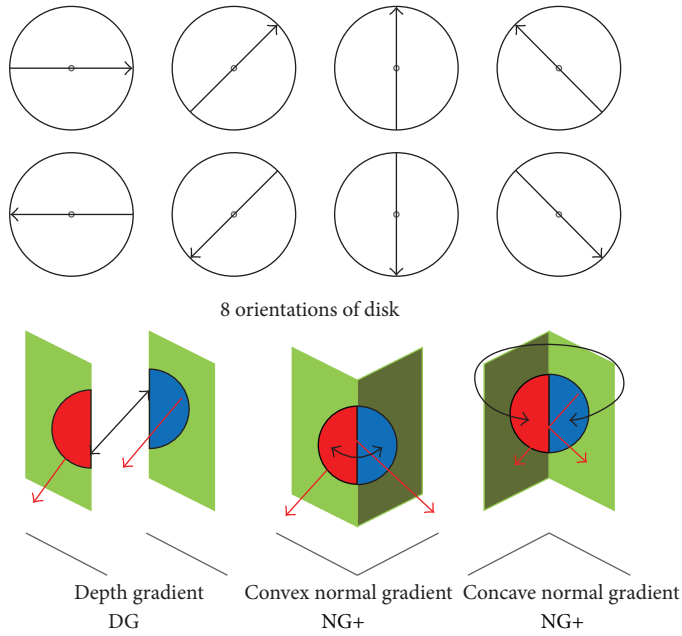

Figure 2: Orientations of disk and geometric gradient.

two halves according to the predefined orientation, and information was compared between two halves. This method was proposed originally by Martin et al. [20] for the problem of edge detection in monocular images. In our experiments, 4 disks with different radius varying from 5 to 20 pixels and 8 orientations are used. Three local geometric gradients DG, NG+, and NG− were computed by the point cloud of 2 halves. First, a planar model is used to represent the distribution of points in each half. Then, the distance between the two planes and the disk center is represented as DG, and the angles between the normal and two planes are, respectively, represented as NG+ and NG−. Figure 2 shows the details.

3. Structured Forests Algorithm

This paper chooses structured random forests as a classifier due to the following four factors. (1) The classification is very fast. The speed of the classifier based on the tree structure is proportional to the depth of the tree, and the depth of the tree is often low. (2) The effect of classification is very good. Neural network adopts probing method to determine the network structure, which leads to the low training efficiency of neural network. Support vector machine (SVM) is to solve two-class problem, but random forest is to solve the multiclass problem. (3) The effect of parallelization is very good. Random forests can be parallel to the different pixel locations and different trees in the scene image, so the efficiency of the calculation is high. (4) The addition of structural information makes the classification more accurate.

The patches in the edges are good to exhibit the local structure of images, such as linear or T-type junctions. Dollar and Zitnick [15] use the structure of the local patches in the image edge to propose an accurate and computationally efficient edge detector. In [21], they improve the algorithm, but partial image details are missed. In this paper, the random decision forest algorithm is applied to a structured learning framework to predict the local edge information. Such a new method for learning decision tree has good robustness, and the structured label of the discrete can estimate standard information gain.

3.1. Random Decision Forest. A decision tree $f_t(x)$ uses recursive method to classify sample x ($x \in X$) into left or right subtree until it reaches a leaf node. In particular, every node j in the tree (each node can be viewed as a weak classifier [22]) is associated with a binary partition function:

$$h\left(x, \theta_j\right) \in \{0, 1\}. \tag{1}$$

If $h(x, \theta_j) = 0$, sample x is classified into the right of the node j or else the left until it reaches a leaf node. After the prediction of the tree, the output of the input x is y ($y \in Y$), which is in the leaf node.

The training of each tree is independent and the recursion is used. For a given node j and training set $S_j \subset X \times Y$, it is important to find a parameter θ_j of the partition function for obtaining a good split of the data:

$$I_j = I\left(S_j, S_j^L, S_j^R\right), \tag{2}$$

where $S_j^L = \{(x, y) \in S_j \mid h(x, \theta_j) = 0\}$ and $S_j^R = S_j \setminus S_j^L$. The norm of selecting the partition parameter θ_j is to maximize information gain I_j. The set S_j^L is used to train the left nodes and the set S_j^R is used to train the right nodes. The train is stopped if one of the following conditions is met. The conditions are as follows. (1) The maximum depth is achieved. (2) Information gain reaches the threshold. (3) The number of samples is less than the threshold value.

Input: training sample set $S_t = \{X, Y\}$, $X = \{x_1, \ldots, x_N\}$, $Y = \{y_1, \ldots, y_N\}$
Output: random tree classifier
if all the training samples of S_t belong to the same category **or** $|S_t| = N \le N_0$, **then**
  return $LeafNode(p_y)$
end if
select parameter space subset randomly: $\Gamma_{\text{sub}}(S_t) \subset \Gamma(S_t)$
for $j = 1$ **to** $|\Gamma_{\text{sub}}(S_t)|$ **do**
  $I_j = H(S_j) - \sum_{k \in \{L,R\}} \frac{|S_j^k|}{|S_j|} H(S_j^k)$
end for
Compute the optimal parameter of the node classifier: θ_j
Set the current dataset of the left and right child nodes $\varnothing$: $S_j^L \leftarrow \varnothing$, $S_j^R \leftarrow \varnothing$
for $i = 1$ **to** N **do**
  if $h(x_i, \theta_j) == 1$ **then**
    $S_L \longleftarrow S_L \cup \{(x_i, y_i)\}$
  else
    $S_R \longleftarrow S_R \cup \{(x_i, y_i)\}$
  end if
end for
New left child node: $LeftNode = GrowRandomizedTree(S_L)$
New right child node: $RightNode = GrowRandomizedTree(S_R)$
return $ParaentNode(\theta, LeftNode, RightNode)$

Algorithm 1: Growth randomized tree (S_t).

Input: Training sample set $S = \{X, Y\}$, $X = \{x_1, \ldots, x_N\}$, $Y = \{y_1, \ldots, y_N\}$
Output: Random Forest Classifier
for $t = 1$ **to** T **do**
  Random sample the training set: $S_t = BootstrapSampling(S)$
  Train the random tree: $TreeRoot_t = GrowRandomizedTree(S_t)$
end for
$RandomForest = \{TreeRoot_1, \ldots, TreeRoot_T\}$
return $RandomForest$

Algorithm 2: Random forest classifier training algorithm.

For multiclass classification, the standard definition of information gain can be defined as

$$I_j\left(\theta_j \mid S\right) = H\left(S_j\right) - \sum_{k \in \{L,R\}} \frac{\left|S_j^k\left(\theta_j\right)\right|}{\left|S_j\right|} H\left(S_j^k\left(\theta_j\right)\right), \quad (3)$$

where $H_{\text{entropy}}(S) = -\sum_y p_y \log(p_y)$ represents Shannon entropy and p_y is the probability of the set S with label y. The Gini impurity $H_{\text{Gini impurity}}(S) = \sum_y p_y(1 - p_y)$ can be used in (3) alternatively.

A decision forest is a collection of T independent decision trees f_t. For a given sample x, it uses decision tree $f_t(x)$ to get an output. The selection of ensemble models mainly depends on the set of outputs Y, which includes average regression, voting mechanism for classification, and more complicated ensemble models [22].

Leaf nodes of the decision tree may store any information. Whether the leaf nodes are reached depends on the input x, so the prediction on multiple trees must rely on the effective integration mode (ensemble model). The output y may be stored in every leaf node, which allows using complicated outputs Y including structural output results in the paper [23].

There may be high variance and overfitting in the single decision tree. In order to reduce high variance of the classifier, the randomness is introduced in the tree's growth process and the training sample of the selection tree. The decision forest trains multiple decision trees without relevance and combines their outputs to improve the disadvantages. The critical part of the training is to achieve the diversity of the decision tree.

In order to guarantee the diversity of the tree, the Bootstrap Sampling is adopted to resample and the samples with differences are generated. At the node level, the randomness is introduced to generate the model of higher accuracy, which is also proved to be very effective. For each node, Γ_{sub} was selected from Γ (complete parameter space) randomly, and Γ_{sub} is a subset of Γ. The algorithm of random forest classifier training is shown in Algorithms 1 and 2.

In practical application, the accuracy of a single decision tree is lost to meet the diversity of the ensemble model.

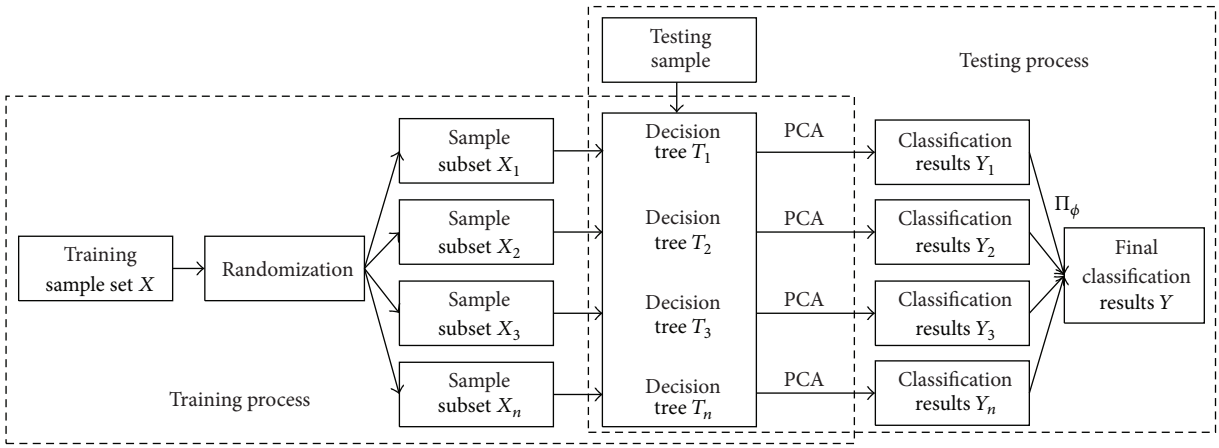

Figure 3: Training process and testing process of structured random forests.

A similar approach is used to introduce approximate information gain norm for the structured labels, that is, the structured decision forest discussed in this paper.

3.2. Structured Random Decision Forests. In this part, the random decision forest is extended to the structured output space Y. There are two main challenges in using structured labels to train random forests. First, structured output space is usually high-dimensional and complicated. Therefore, the computation of evaluation on many structured label splits is very expensive. Second, it is not an accurate method to define the information gain of the structured labels.

In this paper, the approximation measurement of the information gain is used to train the random forest classifier. The optimal split function, which is described above, is not necessary. We aim to map all the structured labels $y \in Y$ at a given node into a discrete set of labels $c \in C$, where $C = \{1, \ldots, k\}$. A similar structured label y is assigned to the same discrete label c.

Calculation of information gain depends on measuring similarity on Y. However, for most structured output spaces including those used for edge detection, the similarity calculation on Y is not easy to define. Therefore, a mapping from Y to temporary space Z is defined, and the distance of the space Z is easy to measure. Finally, the two steps are as follows: (1) mapping $Y \rightarrow Z$ (2) mapping $Z \rightarrow C$. We will describe the method in detail in the next installment.

For most structured output spaces including those used for edge detection, a mapping form is defined as

$$\Pi : Y \longrightarrow Z. \tag{4}$$

Thus we can approximate estimation dissimilarity of $y \in Y$ by the Euclidean distance over Z.

The space Z may be high-dimensional. So we resample Z at m dimensions and get a reduced mapping $\Pi_\phi : Y \rightarrow Z$. During training, different mappings Π_ϕ are randomly generated and applied to training labels Y_j at each node j. This method has two advantages: (1) computing Π_ϕ is much faster than computing Π; (2) extra randomness is introduced during resampling Z, which guarantees the diversity of the tree.

We also introduced Principal Component Analysis to reduce the dimensionality of Z. PCA not only can denoise Z, but also approximately keep the Euclidean distance unchanged. In our experiment, when Π_ϕ is used with $m = 256$, the dimensionality processed by PCA mapping is at most 5 dimensions.

There are lots of possibilities to calculate the information gain while giving a mapping: $\Pi_\phi : Y \rightarrow Z$. We introduced a simple and efficient method. Map a set of structured labels $y \in Y$ into other discrete labels $c \in C$, where $C = \{1, \ldots, k\}$. The labels similar to z are assigned to the same discrete label c. Discrete label may be dualistic ($k = 2$) or multivariate ($k > 2$) and we can use the standard information gain as defined in (3) based on Gini impurity or Shannon entropy. It is important to add the discretization separately in training each node and also depend on the distribution of the label at a particular node (different from [18]).

There are two methods for mapping a given Z into discrete label C. One approach is to use K-means to cluster z into k clusters. Another approach is to quantize z by PCA based on $\log_2(k)$ dimensions and assign a discrete label c according to the quadrant of z. The two methods are similar, but the latter is faster. We select PCA with $k = 2$ in the experiment.

Finally, how to combine n labels ($y_1 \cdots y_n \in Y$) into a single forecasting mechanism is defined. This mechanism can accomplish training (correlating labels to nodes) and testing (merging multiple predictions) simultaneously. Similarly, we use m-dimensional mapping Π_ϕ to compute $z_i = \Pi_\phi(y_i)$ for each label i. The labels y_j ($j = 1 \cdots n$) with z_k-centered are chosen, which make the sum of distance between z_k and z_i ($i = 1 \cdots n$) minimize. Figure 3 shows training process and testing process of structured random forests.

4. Edge Extraction from RGB-D Image

This part discusses how to apply the feature information to the structure forest model and extract edges from a

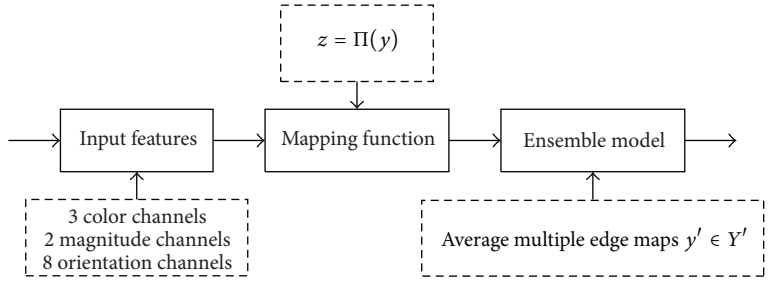

Figure 4: Process flow of edge extraction.

RGB-D image using this system. The system's input image will contain a variety of information, for example, RGB or RGB-D. According to this information, the dualistic variable is used to indicate whether it has edges or not. The method of semantic image labeling is similar to that in the paper [23]. The label in a small image block is independent of each other, which provides a good candidate for the structure forest approach.

Given a segmented training image sample, the edge between the patches is regarded as edge. An image patch is segment membership for each pixel with segmentation mask indicator or a dualistic edge map. The former is expressed as $y = Y = Z^{d\times d}$, and the latter is expressed as $y' \in Y' = \{0, 1\}^{d\times d}$, where d is means of patch width. An edge map y' is deduced from segmentation mask y. Two representations are both used in this paper. The main process flow of our edge extraction algorithm for RGB-D image is as shown in Figure 4.

How to compute input features x and use x for segmentation mapping functions Π_ϕ is discussed in the remainder of this section. In addition, we introduce integrate multiple prediction ensemble models.

A structured 16×16 segmentation mask is predicted from a large 32×32 patch by our method. For each image patch, we add additional information channels; thus a feature vector $x \in R^{32\times32\times K}$ is generated, where K is the number of channels. We use them in pixel lookups $x(i, j, k)$ and pairwise differences $x(i_1, j_1, k) - x(i_2, j_2, k)$.

Inspired by the edge detection algorithm proposed by Lim et al. [18], a similar set of color and gradient channels are used in our paper. We compute 3-channel color information and gradients with 2 normalized scales (original and half resolution) in LUV color space. In addition, the gradient channels of each scale are divided into 4 channels based on the directions. We use the triangle filter with the radius of 2 and the downsampling with coefficient of 2. So the final result is 3 kinds of color information, 2 kinds of magnitude, and 8 directions for a total of 13 channels information.

Because the factor of the downsampling is 2, there are $32 \cdot 32 \cdot 13/4 = 3328$ candidate feature points $x(i, j, k)$. Then we compute the difference between the different feature points. For every channel (8-pixel radius), triangular fuzzy is applied and the resolution is 5×5 by the downsampling. We sample all the candidate pairs and compute the difference. Finally, there are $\binom{5\cdot5}{2} = 300$ candidate features for each channel and 7228 candidate features for each patch.

In order to train decision trees, a mapping $\Pi : Y \to Z$ needs to be defined. Because we choose a structured label y of 16×16 segmentation masks, one choice is to use $\Pi : Y \to Y'$, where y' is dualistic edge map corresponding to y. But the Euclidean distance of space Y' cannot be computed.

We define another mapping Π, using $y(j)$ $(1 \le j \le 256)$, to denote the jth pixel of mask y. Because the definition of y is only related to the permutation, the generation of $y(j)$ has no information about y. We can check if $y(j_1)$ is equal to $y(j_2)$ with $j_1 \ne j_2$. In summary, a large-scale mapping function $z = \Pi(y)$ is defined, which encodes $[y(j_1) = y(j_2)]$ for every pair of feature points with $j_1 \ne j_2$. If the dimension of Z is $\binom{256}{2}$, we need only to compute an image patch with the subset of m dimensions. We get a better result and capture the similar segmentation masks with $m = 256$ and $k = 2$.

The output result of random forest is more robust by combining outputs of the decorrelated decision trees. However, it is very difficult to achieve the fusion of multiple segmentation masks $(y \in Y)$. Generally, multiple boundary mapping $(y' \in Y')$ is used to achieve indirectly the fusion. This paper takes advantage of the function that decision tree leaf node is capable of storing any information. Besides segmentation mask y, this paper learns corresponding boundary mapping y', which enable multiple predictive results of the decision tree to fuse averagely.

We use structured labels to capture entire image information and reduce the number of decision trees T which are used to evaluate each pixel. So our method is very efficient. The structured output is calculated with the density of 2 pixels. For 16×16 image patches, it receives $16^2T/4 \approx 64T$ predictions per pixel. In experiment, we set $1 \le T \le 4$.

5. Experiments

The images of our experiment are captured by Xbox Kinect 360 in the laboratory. We work with the NYUD2 dataset and use the standard split of 795 training images and 654 testing images. These splits are carefully selected such that images from the same scene are only in one of these sets. Experimental results for laboratory scenes are shown in Figures 5 and 6.

In Figures 5 and 6, RGB images are in the first row and depth images are in the second row. The third row is the algorithm proposed by Gupta et al. [12], who use SVMs as classifier and combine the algorithm of depth image edge detection. This algorithm is referred to as SG algorithm

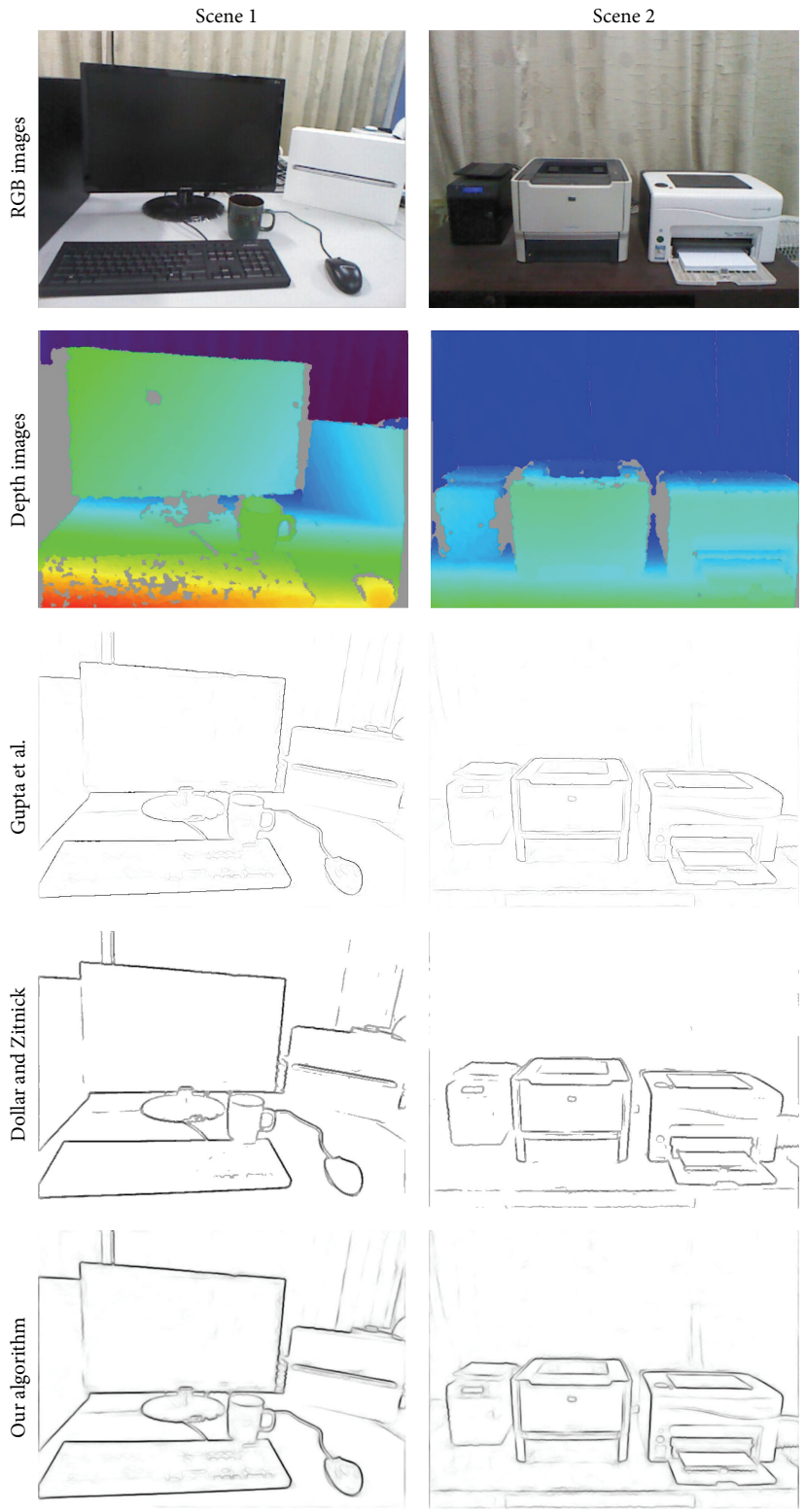

Figure 5: Comparison of three algorithms in different scene.

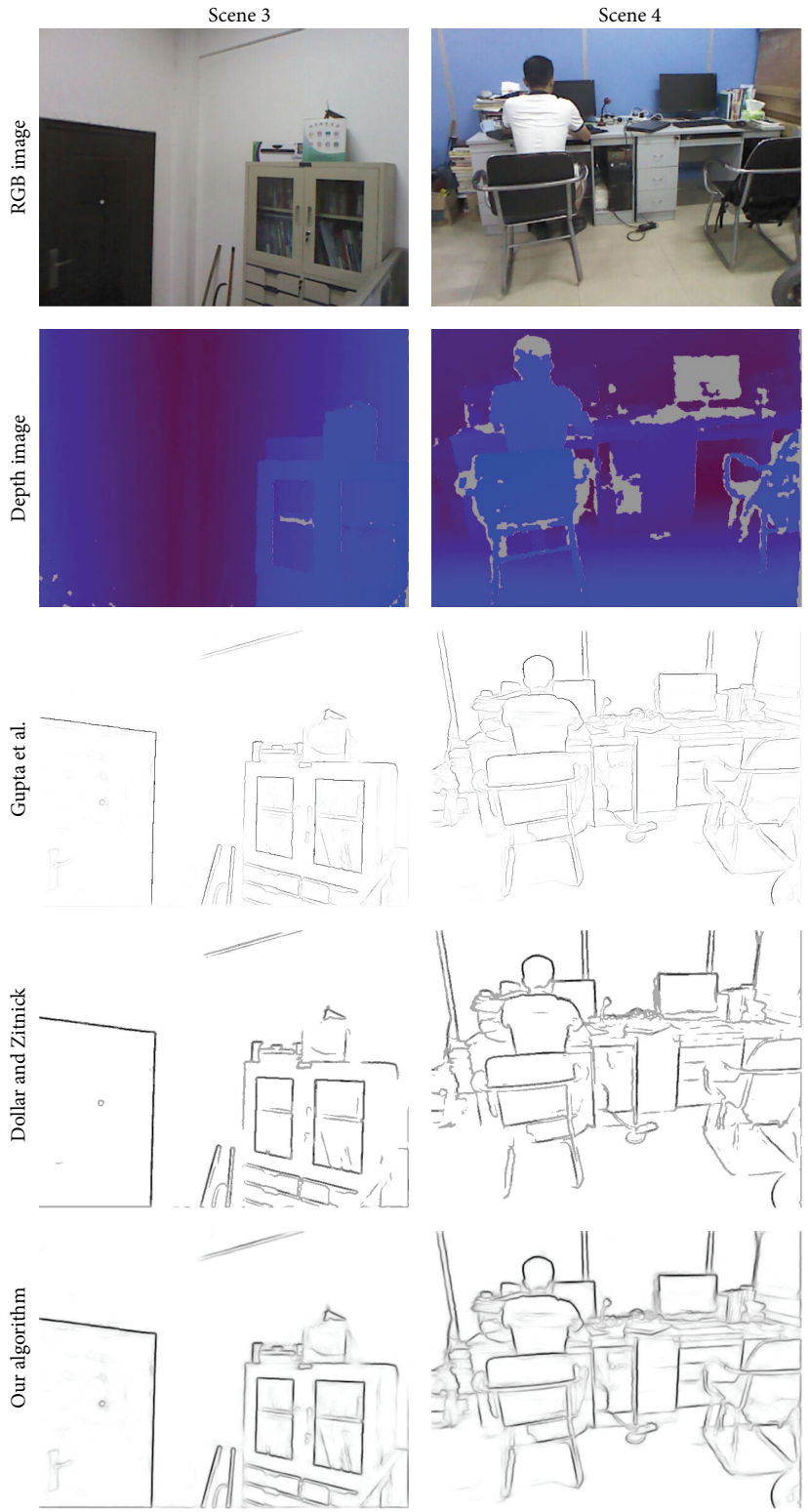

Figure 6: Comparison of three algorithms in different scene.

below. The fourth row is the algorithm proposed by Dollar and Zitnick [15], who use structured forests method. This algorithm is referred to as PD algorithm below. The fifth row is our method.

In experiment, we choose four different scenes as the detection object, and the details of the four scenes were different. As shown in Figures 5 and 6, from scene 1 to scene 4, the details of the image are increasing. According to the above results, we can find that, in the results of SG algorithm, the lost edge information is too much, and the details of the image nearly cannot be recognized. For example, in scene 1, the edge of the display and the line of keyboard and mouse are not clear; the keys of keyboard cannot be identified. From the details of the image, PD algorithm and our method are significantly better than SG algorithm.

According to the experimental results, the results from algorithm proposed by Piotr Dollar mostly can be observed. In the scene of the image details are less such as scenes 1 and 2; the performance is good. When the details of the image are more such as scenes 3 and 4, the performance is poor, the extracted image edge appears fuzzy, some of the boundary is blurred, and the image details are lost more. For example, the handle of drawer in scene 3 is missing. In scene 4, the keyboard is lost, and the outline of the book is not clear.

According to Figures 5 and 6, this paper presents the results of the improved algorithm, the image edge is clear and accurate, and the details of the image are kept well. When the details of the image are less, the performance of our method is almost as good as PD algorithm. When the details of the image are increased, our method is better than PD algorithm. In particular, in the last scene, we propose an improved algorithm that is significantly better than PD algorithm in the presentation of the details of the desktop.

To quantitatively compare our algorithm with others we work with the NYUD2 dataset. We report the standard maximum F-measure, that is, $F_{\max}$, in Table 1. F-measure is the harmonic mean of precision and recall traditionally, which is given by

$$F_\beta = \frac{\left(\beta^2 + 1\right) \text{Precision} \cdot \text{Recall}}{\beta^2 \text{Precision} + \text{Recall}}. \quad (5)$$

When $\beta = 1$, recall and precision are evenly weighted and F_1 measure is defined by

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \quad (6)$$

Precision is the number of correct positive results divided by the number of all positive results, and Recall is the number of correct positive results divided by the number of positive results that should have been returned. F-measure can be interpreted as a weighted average of Precision and Recall, and it can reflect the overall indicators.

We plot the precision-recall curve on edge in Figure 7 and report the standard maximum F-measure metric ($F_{\max}$) in Table 1. As shown in Table 1, in our experiments, we report three quantities for an algorithm, the Optimal Dataset Scale (ODS) or best F-measure on the dataset for a fixed scale, the Optimal Image Scale (OIS) or aggregate F-measure on the dataset for the best scale in each image, and the Average Precision (AP) on the full recall range.

Table 1: Edge benchmarks on NYUD2.

	ODS ($F_{\max}$)	OIS ($F_{\max}$)	AP
Gupta et al.	68.79%	71.35%	63.37%
Dollar and Zitnick	68.13%	69.98%	68.41%
Our algorithm	70.03%	72.11%	69.43%

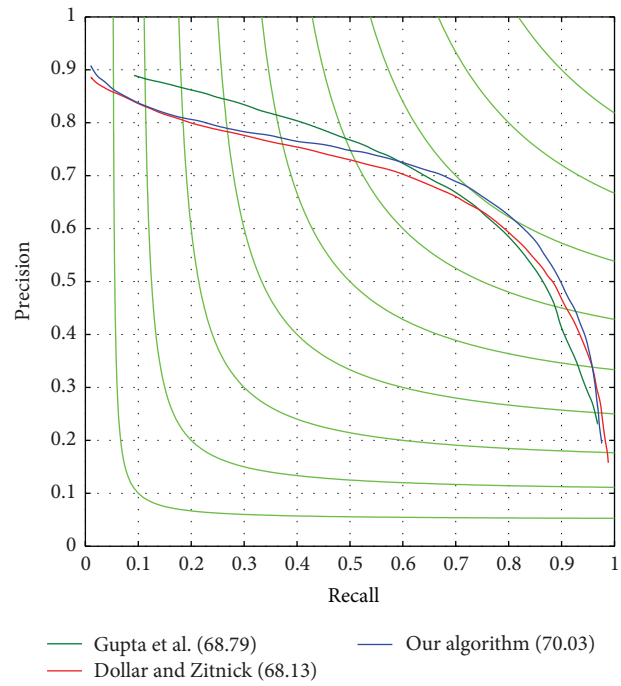

Figure 7: Evaluation of edge detectors on NYUD2 benchmark.

As shown previously, Dollar and Zitnick proposed a novel learning approach based on structured random forests to classify a pixel as a contour pixel or not. However, their approach treats the depth information as another image, rather than encoding it in terms of geocentric quantities. Gupta et al. encode depth information in terms of geocentric quantities, like NG−, making full utilization of the information of RGB-D image. We extract the advantages of the two methods to obtain a new algorithm. As shown in Figure 4, according to precision-recall curve, in our method, regardless of the size of the recall value, precision will get a relatively stable value. For Gupta et al., when the precision value is larger, the value of recall is too small. For Dollar and Zitnick, when the recall value is larger, the value of precision is too small.

6. Conclusions

This paper exploits the information of RGB-D images and makes full use of the brightness, color, texture, and depth information by means of a circular disc. Before classification, the adaptive bilateral filter is used to deal with the depth

image and the output of the random decision forest is structured, which makes the classification more accurate. Experimental results show that the proposed algorithm is more accurate than the others, and the algorithm can also show a high degree of accuracy in the scene where image details are rich. But in the experiment, we also found that the speed of the improved algorithm is not accelerated, the same as the SG algorithm and PD algorithm. The next step is to ensure the accuracy of the algorithm to improve the speed of the algorithm.

Competing Interests

The authors declare that there is no conflict of interests with any company or organization regarding the material discussed in this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grants nos. 61563014 and 61165007) and the Natural Science Foundation of Jiangxi Province (no. 20161BAB202068).

References

[1] N. Silberman, L. Shapira, R. Gal, and P. Kohli, "A contour completion model for augmenting surface reconstructions," in *Proceedings of the 13th European Conference on Computer Vision, Part III*, pp. 488–503, Zurich, Switzerland, 2014.

[2] I. E. Sobel, *Camera models and machine perception [Ph.D. thesis]*, Stanford University, Stanford, Calif, USA, 1970.

[3] J. M. Prewitt, "Object enhancement and extraction," *Picture Processing and Psychopictorics*, vol. 10, pp. 15–19, 1970.

[4] R. A. Kirsch, "Computer determination of the constituent structure of biological images," *Computers and Biomedical Research*, vol. 4, no. 3, pp. 315–328, 1971.

[5] L. G. Roberts, "Machine perception of three dimensional solids," in *Optical and Electro-Optical Information Processing*, J. T. Tippet, Ed., MIT Press, Cambridge, Mass, USA, 1965.

[6] D. Marr and E. Hildreth, "Theory of edge detection," *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 207, pp. 187–217, 1980.

[7] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[8] S. M. Smith and J. M. Brady, "SUSAN—a new approach to low level image processing," *International Journal of Computer Vision*, vol. 23, no. 1, pp. 45–78, 1997.

[9] J. Scharcanski and A. N. Venetsanopoulos, "Edge detection of color images using directional operators," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 2, pp. 397–401, 1997.

[10] S. Di Zenzo, "A note on the gradient of a multi-image," *Computer Vision, Graphics and Image Processing*, vol. 33, no. 1, pp. 116–125, 1986.

[11] C. Cai and S. K. Mitra, "A normalized color difference edge detector based on quaternion representation," in *Proceedings of the International Conference on Image Processing (ICIP '00)*, pp. 816–819, Vancouver, Canada, September 2000.

[12] S. Gupta, P. Arbelaez, and J. Malik, "Perceptual organization and recognition of indoor scenes from RGB-D images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 564–571, Portland, Ore, USA, June 2013.

[13] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898–916, 2011.

[14] S. Maji, A. C. Berg, and J. Malik, "Efficient classification for additive kernel SVMs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 66–77, 2013.

[15] P. Dollar and C. L. Zitnick, "Structured forests for fast edge detection," in *Proceedings of the 14th IEEE International Conference on Computer Vision (ICCV '13)*, pp. 1841–1848, Sydney, Australia, December 2013.

[16] J. Zhou, J. Yan, T. Wei, K. Wu, X. Chen, and S. Hu, "Sharp corner/edge recognition in domestic environments using RGB-D camera systems," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 10, pp. 987–991, 2015.

[17] S. Sarkar, V. Venugopalan, K. Reddy, M. Giering, J. Ryde, and N. Jaitly, "Occlusion edge detection in RGB-D frames using deep convolutional networks," http://arxiv.org/abs/1412.7007.

[18] J. J. Lim, C. L. Zitnick, and P. Dollar, "Sketch tokens: a learned mid-level representation for contour and object detection," in *Proceedings of the 26th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '13)*, pp. 3158–3165, IEEE, Portland, Ore, USA, June 2013.

[19] A. Savitzky and M. J. E. Golay, "Smoothing and differentiation of data by simplified least squares procedures," *Analytical Chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964.

[20] D. R. Martin, C. C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 530–549, 2004.

[21] P. Dollár and C. L. Zitnick, "Fast edge detection using structured forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 8, pp. 1558–1570, 2015.

[22] A. Criminisi, J. Shotton, and E. Konukoglu, "Decision forests: a unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning," *Foundations and Trends in Computer Graphics and Vision*, vol. 7, no. 2-3, pp. 81–227, 2011.

[23] P. Kontschieder, S. R. Bulò, H. Bischof, and M. Pelillo, "Structured class-labels in random forests for semantic image labelling," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV '11)*, pp. 2190–2197, Barcelona, Spain, November 2011.