sensors

MDPI

*Article*

# Real-Time 3D Reconstruction Method Based on Monocular Vision

**Qingyu Jia [1], Liang Chang [1], Baohua Qiang [1,*], Shihao Zhang [1], Wu Xie [1], Xianyi Yang [1,*], Yangchang Sun [2] and Minghao Yang [2]**

[1] Guangxi Key Laboratory of Image and Graphics Intelligent Processing, Guilin University of Electronic Technology, Guilin 541004, China; jqy_kin@163.com (Q.J.); changl@guet.edu.cn (L.C.); shihao_zhang@yeah.net (S.Z.); xiewu588@guet.edu.cn (W.X.)

[2] Research Center for Brain-Inspired Intelligence (BII), Institute of Automation, Chinese Academy of Sciences (CASIA), Beijing 100190, China; sunyangchang2020@ia.ac.cn (Y.S.); mhyang@nlpr.ia.ac.cn (M.Y.)

* Correspondence: qiangbh@guet.edu.cn (B.Q.); xianyiyang65@126.com (X.Y.)

**Abstract:** Real-time 3D reconstruction is one of the current popular research directions of computer vision, and it has become the core technology in the fields of virtual reality, industrialized automatic systems, and mobile robot path planning. Currently, there are three main problems in the real-time 3D reconstruction field. Firstly, it is expensive. It requires more varied sensors, so it is less convenient. Secondly, the reconstruction speed is slow, and the 3D model cannot be established accurately in real time. Thirdly, the reconstruction error is large, which cannot meet the requirements of scenes with accuracy. For this reason, we propose a real-time 3D reconstruction method based on monocular vision in this paper. Firstly, a single RGB-D camera is used to collect visual information in real time, and the YOLACT++ network is used to identify and segment the visual information to extract part of the important visual information. Secondly, we combine the three stages of depth recovery, depth optimization, and deep fusion to propose a three-dimensional position estimation method based on deep learning for joint coding of visual information. It can reduce the depth error caused by the depth measurement process, and the accurate 3D point values of the segmented image can be obtained directly. Finally, we propose a method based on the limited outlier adjustment of the cluster center distance to optimize the three-dimensional point values obtained above. It improves the real-time reconstruction accuracy and obtains the three-dimensional model of the object in real time. Experimental results show that this method only needs a single RGB-D camera, which is not only low cost and convenient to use, but also significantly improves the speed and accuracy of 3D reconstruction.

**Keywords:** monocular vision; YOLACT++; deep optimization; real-time 3D reconstruction



**Citation:** Jia, Q.; Chang, L.; Qiang, B.; Zhang, S.; Xie, W.; Yang, X.; Sun, Y.; Yang, M. Real-Time 3D Reconstruction Method Based on Monocular Vision. *Sensors* **2021**, *21*, 5909. https://doi.org/10.3390/s21175909

Academic Editor: Simon X. Yang

Received: 21 July 2021
Accepted: 29 August 2021
Published: 2 September 2021



## 1. Introduction

Real-time 3D reconstruction technology is not only a scientific problem that has been widely studied, but also the core technology of many applications, including robotics, virtual reality, digital media, human–computer interaction, intangible cultural heritage protection, etc.

Traditional real-time 3D reconstruction methods mainly rely on ordinary RGB cameras to collect images. Dalit Caspi proposed a method that can adapt the projection mode and number according to the characteristics of the scene. It can reduce the number of projection patterns to the necessary limit to achieve the accuracy and robustness of the Gray code technology. It uses composite structured light to complete reconstruction. In addition, Thomas P. Koninckx et al. proposed an adaptive structured light method [1], which combines geometric coding and color coding to balance reconstruction speed and quality through weights. M. Takeda first proposed the Fourier Contour Transformation (FTP) method to obtain the depth information of an object. On this basis, the literature completed

the three-dimensional real-time reconstruction of the surface of complex objects [2,3]. However, such methods not only require complex and expensive hardware facilities, which lack convenience, but also have low reconstruction accuracy and poor realism of the three-dimensional model.

In recent years, with the emergence of different types of depth camera sensors, a more convenient and economical research method has been provided for the research of 3D reconstruction. More and more real-time 3D reconstruction technologies based on depth cameras have been continuously proposed [2–10]. Henry et al. adopted a RGB-D closest point iteration method that makes full use of color information and depth information [11], and proposed a framework called RGB-D Mapping. In this framework, they studied how to use RGB-D cameras to build dense 3D maps of indoor environments. By combining vision and depth information for view-based closed-loop detection, pose optimization is performed to achieve a globally consistent map. On this basis, Izadi et al. designed an interactive 3D reconstruction system KinectFusion, which allows users to hold the Kinect camera to move, and only use the depth information collected by the Kinect camera to track the 3D posture of the sensor. It uses the depth data obtained by the sensor in real time to create accurate geometrical 3D models. Vogiatzis et al. modeled the real-time 3D reconstruction problem from a probabilistic perspective [12], and proposed a novel video based MVS algorithm, which is suitable for real-time interactive 3D modeling. The key idea is the per-pixel probability depth estimation scheme, which updates the posterior depth distribution in each new frame and introduces the idea of SLAM (Simultaneous Localization and Mapping) to solve the real-time 3D reconstruction problem [13,14]. Furukawa et al. proposed a new multi-view stereo vision algorithm for 3D reconstruction [15]. The algorithm outputs a set of dense small rectangular blocks which is covering the visible surface in the image. The key to the performance of the proposed algorithm is an effective technique that enforces local luminosity consistency and global visibility constraints. It is also a concise effective method to convert the generated patch model into a grid, which can be further refined by an algorithm that enforces photometric consistency and regularization constraints. It automatically detects and discards outliers and obstacles, and does not require any initialization in the form of visual hulls, bounding boxes, or effective depth ranges. Many researchers have improved this method on this basis [16]. The above method is very computationally intensive and can only be run in an offline mode. However, its theory has great reference value for real-time 3D reconstruction research. Newcombe et al. proposed a real-time 3D reconstruction system based on multi-view constraints [17]. It is a system for real-time camera tracking and reconstruction. It does not rely on feature extraction, but on dense per-pixel methods. When a single handheld RGB camera flies over a static scene, a detailed texture depth map of selected key frames is estimated to generate a patchwork of surfaces with millions of vertices. Aiming at the 3D structure restoration process in real-time 3D reconstruction problems, they proposed a Cost Volume-based 3D structure Recovery method [18,19]. Yang et al. developed a real-time 3D reconstruction system suitable for UAVs on Jetson TX2 [20]. Jetson TX2 is an embedded computer developed by NVIDIA. This system uses a GPU-accelerated semi-global method to reconstruct the three-dimensional scene taken by the drone. In addition, Schöps et al. developed a real-time 3D reconstruction system on the Jetson TX1 embedded computer [21]. This system achieves a depth estimation method based on multi-resolution, by considering two different perspectives of the original image and after down sampling. The constrained relationship between image pixels reconstructs a three-dimensional model of the scene. In addition, the silhouette-based method is often used for real-time 3D reconstruction [22,23]. It can extract the contour of the foreground object in each image and reconstruct the 3D model. Although their time complexity is low, they cannot reconstruct details, especially for concave surfaces. We need to extract the contours of objects robustly. Tong et al. proposed an efficient system that uses three Kinects and one rotating stage to scan and generate models offline [24]. First, a rough mesh template is constructed and used to deform successive frames pairwisely. Second, global alignment is performed to distribute errors in the

deformation space, which can solve the loop closure problem efficiently [25]. On this basis, Maimone et al. proposed an efficient system with six Kinects [26], which combines the individual three-dimensional grids from each Kinect only in the rendering stage to produce an intermediate stereoscopic view. It presents a high quality effect. The 3D reconstruction method proposed by Alexiadis et al. uses multiple Kinects to generate a single 3D mesh model in real time [27]. In order to further solve the problem of non-smooth surface and holes in the reconstructed 3D mesh model [28], Alexiadis et al. proposed a volumetric method, which implements real-time 3D reconstruction by parallelizing it on a graphics processing unit [29].

Although the above methods are mainly aimed at real-time 3D reconstruction, they still have the following shortcomings:

- There are many cameras and various sensors required for reconstruction, which are expensive and poor in portability.
- The reconstruction speed is slow. The reconstruction method is computationally expensive and time-consuming. It cannot meet real-time requirements.
- The reconstruction error is large, especially for the depth error. The reconstruction model effect is poor.

In order to solve the above problems, in this research, we only use a single depth camera and propose a real-time 3D reconstruction method, which can quickly and accurately obtain a 3D model of the scene. The contributions of this research are as follows:

- We use a single depth camera for real-time collection of visual information, and use the powerful real-time segmentation capabilities of the YOLACT++ network to extract the real-time collected information [30–32]. Only part of the information of the extracted items is reconstructed to ensure the real-time performance of the method.
- We propose a visual information joint coding three-dimensional restoration method (VJTR) based on deep learning [33,34]. This method combines three stages of deep recovery, deep optimization, and deep fusion. Taking advantage of the high accuracy and fast running speed of ResNet-152 network, through joint coding of different types of visual information, the three-dimensional point cloud with optimized depth value corresponding to the two-dimensional image is output in real time [35–40]. At the same time, the most critical visual information combination in the process of this method is determined, so as to ensure the accuracy of the reconstruction of the method.
- For the outliers generated in the process of reconstructing the scene, we propose an outlier adjustment method based on cluster center distance constrained (BCC-Drop) to ensure the reconstruction of the space of each object consistency and reconstruction accuracy.
- We propose a framework organization method that can use a single depth camera to quickly and accurately perform 3D reconstruction. It is without any human assistance or calibration, and can automatically organize the reconstructed objects in the 3D space.
- Experimental results show that our method greatly improves the performance of 3D reconstruction and is always better than other mainstream comparison methods. The reconstruction speed reaches real-time and can be used for real-time reconstruction.

The rest of the paper is organized as follows: the proposed method is explained in Section 2, the experiment and results are shown in Section 3, and the conclusions are drawn in Section 4.

## 2. Materials and Methods

### *2.1. Framework*

Our proposed method is shown in Figure 1. The framework consists of three steps. Firstly, a single RGB-D camera is used to collect visual information, and the YOLACT++ network is used to segment the input image to extract the visual information that needs

to be reconstructed. Secondly, for the segmented visual information, the VJTR method is used to obtain the three-dimensional point information of the object. The detailed network structure diagram of the YOLACT++ and VJTR will be given as Figures 1 and 2 in the following chapters. Finally, we propose an outlier adjustment method based on cluster center distance constrained (BCC-Drop) to remove the outliers in the reconstruction process, which is used to reduce the object reconstruction error and improve the 3D reconstruction accuracy. We save each 3D point generated through the VJTR network as a one-dimensional array, obtain the color information of the point from the RGB image according to the two-dimensional pixel value of the point, and store the color information of the 3D point and the 3D point in the same array. The 3D point set is visualized by OpenGL, and the fuzzy reconstruction image is obtained. By processing the 3D point set generated by the VJTR network with the BCC-Drop method, accurate 3D point set information is obtained. Then, the accurate three-dimensional point set information is reconstructed by the above method to obtain a clear reconstruction image. Experimental results show that this method only needs a single RGB-D camera, which is not only low cost and convenient to use, but also significantly improves the speed and accuracy of 3D reconstruction.

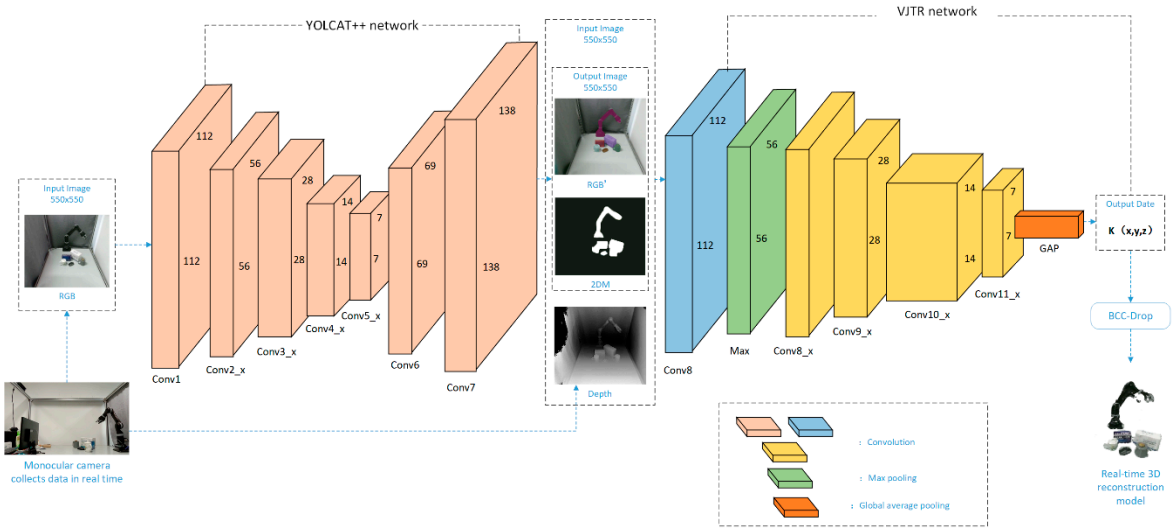


**Figure 1.** Flow chart of the method.

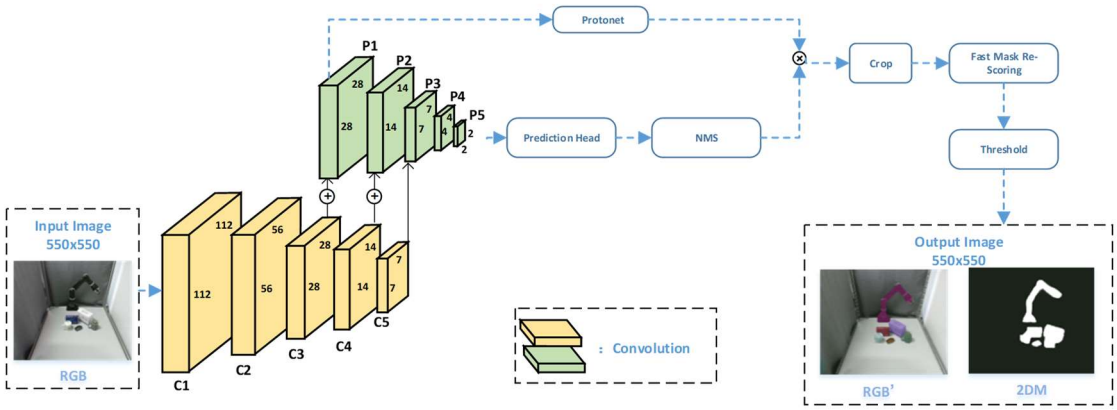


**Figure 2.** The network structure of YOLACT++.

### 2.2. Visual Information Segmentation and Extraction

We use the YOLACT++ network to segment the visual information collected by the RGB-D camera in real time. As shown in Figure 2, the RGB image collected in real time is

passed to the YOLACT++ network, which is used for information extraction, to obtain the segmented image RGB'.

The input image size of the YOLACT++ model is 550 × 550 and the Backbone used is ResNet101. Five convolution modules of the ResNet101 are used in YOLACT++. They are conv1, conv2_x to conv5_x, corresponding to C1, C2 to C5 in Figure 2. For each standard 3 × 3 convolutional layer in C3–C5, in order to achieve trade-off in terms of time and performance, we replace it with variability convolution every three convolutional layers [41].

P1 to P5 is the FPN network in Figure 2. It obtains P3 from C5 first through a convolutional layer, then uses bilinear interpolation on P3 to double the feature map, and adds it to the convolutional C4 to obtain P2. Then, we used the same method to find P1. We convolved and down sampled P3 to find P4, and performed the same convolution and down sampling on P4 to find P5, thereby establishing an FPN network. The advantage of using FPN is that it can enrich the features learned by the model. The next step is the parallel operation. P1 is sent to Protonet, and P1–P5 are also sent to Prediction Head at the same time. The Protonet network module is composed of several convolutional layers. Its input is P1, and its output mask dimension is $138 \times 138 \times k(k = 32)$, that is, 32 prototype masks with the size of 138 × 138. The branch of Prediction Head is improved on the basis of RetinaNet [42]. It uses a shared convolutional network which can increase the speed and achieve the purpose of real-time segmentation. Its input is a total of five feature maps P1–P5. Each feature map generates anchors first. Each pixel generates 3 anchors with ratios of 1:1, 1:2, and 2:1. The anchor basic side lengths of the five feature maps are 24, 48, 96, 192, and 384, respectively. The basic side length is adjusted according to different proportions to ensure that the area of the anchor is equal.

The mask coefficient obtained by the prediction head and the prototype mask obtained by the protonet are subjected to matrix multiplication to obtain the mask of each target object in the image. In the second half of the model, a mask re-scoring branch was added. This branch uses the cropped prototype mask (non-threshold) generated by YOLACT as input, and outputs the IoU corresponding to each category of the GT mask. The fast mask re-scoring branch consists of 6 convolutional layers and 1 global mean pooling layer. Crop refers to clearing the mask outside the boundary. The boundary of the training phase is the ground truth bounding box, and the boundary of the evaluation phase is the predicted bounding box. Threshold refers to the image binarization of the generated mask with a threshold of 0.5.

### *2.3. Visual Information Joint Coding Three-Dimensional Restoration Method*

After we obtain the segmented visual information, as the segmentation is performed along the contour of the object edge, the depth value of the contour edge is prone to large errors at this time. We first introduce the mathematical representation of the process of this method from the RGB image and the Depth image, respectively. Then, we introduce how to use the convolutional neural network (CNN) to obtain the estimated value of the three-dimensional position from the visual information of a single RGB-D camera. We call 3D point reconstruction using a convolutional neural network as VJTR.

#### 2.3.1. Reconstruction of 3D Coordinates from RGB Image and Depth Image

Assuming that a certain pixel in a two-dimensional RGB image is represented by $I_I = [u, v, 1]^T$, and its corresponding three-dimensional reconstruction point is represented by $W_I = [X_I, Y_I, Z_I]^T$, the process of mapping RGB image pixels to three-dimensional reconstruction positions is described as

$$\tau I_I = F_I W_I, \text{ where } F_I = \begin{bmatrix} f_x & \eta & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (1)$$

In Equation (1), $\tau$ is an arbitrary scale factor. $F_I$ is the intrinsic matrix of the camera, $u_0$ and $v_0$ are the coordinates in the image coordinate system. $f_x$ and $f_y$ are the image scale factors along the $u$ and $v$ axes, respectively, and $\eta$ is the skew parameter of the image between $u$ and $v$.

Equation (1) can be further simplified as

$$I_I = M_I W_I, \text{ where } M_I = \frac{1}{\tau} F_I. \tag{2}$$

Setting the U, V value of pixel I of RGB image and the camera intrinsic matrix to $F_I$, we can use Equation (2) to calculate the $X_I$, $Y_I$ mapped to the three-dimensional space from a single RGB image. However, for $Z_I$, the true distance from the camera is still uncertain.

Similarly, the relationship between 3D reconstruction point $W_D = [X_D, Y_D, Z_D]^T$ and Depth image projection $I_D = [\alpha, \beta, \gamma]^T$ is

$$\tau I_D = F_D W_D, \text{ where } F_D = \begin{bmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{bmatrix}. \tag{3}$$

$\tau$ is an arbitrary scale factor, $F_D$ is the sensor parameter of the Depth image given by the $3 \times 3$ matrix, which is different from the RGB imaging pinhole model and cannot be directly calculated by the geometric method.

Equation (3) can be further simplified as

$$I_D = M_D W_D, \text{ where } M_D = \frac{1}{\tau} F_D. \tag{4}$$

Combining Equations (2) and (4) together, we can obtain

$$\begin{bmatrix} I_I \\ I_D \end{bmatrix} = \begin{bmatrix} M_I & 0 \\ 0 & M_D \end{bmatrix} \begin{bmatrix} W_I \\ W_D \end{bmatrix}. \tag{5}$$

That is, the three-dimensional position of the pixel can be estimated synchronously from its RGB and Depth images.

Equation (5) can be simply written as

$$I = MW, \tag{6}$$

where $I = [I_I, I_D]^T$, $W = [W_I, W_D]^T$. The $M$ is a diagonal matrix constructed by $M_I$ and $M_D$, $W$ could be obtained from $M^{-1}I$.

#### 2.3.2. Simultaneous Estimation of Three-Dimensional Values Using ResNet-152 Network

High-dimensional image data can be presented by low-dimensional code using deep architecture or CNN-based autoencoder. Similarly, an autoencoder can be used to represent a two-dimensional point from the position of the white pixel embedded in the black background image. In this article, we refer to this image as a two-dimensional position mask (2DM) image. The second row of image is samples of some 2DM images in Figure 3. In this way, the CNN network provides a solution to reconstruct the three-dimensional position of points from the two-dimensional description of RGB images, grayscale Depth images, and 2DM images.

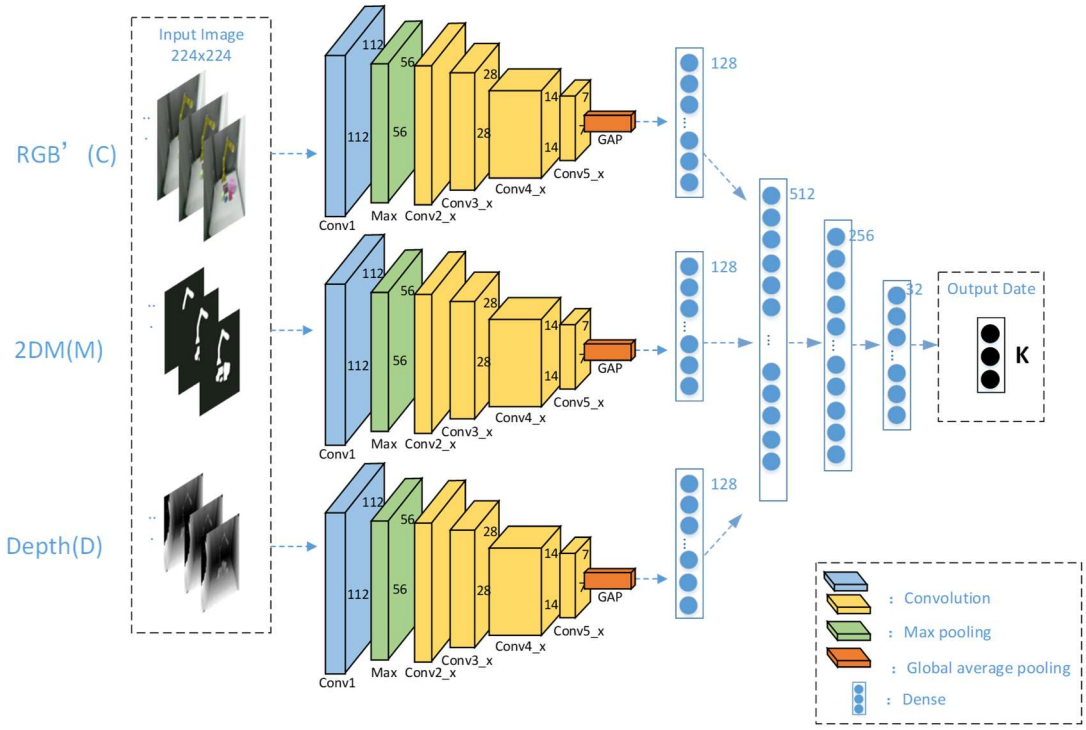


**Figure 3.** Schematic diagram of VJTR method.

In Equation (6), the symbol $I$ contains the 2D point in RGB image and D in depth image. Supposing an auto-encoder operator $A$ transfer $I$ to $P$ with $P = AI$, where the number of $A$'s columns equal to that of $I$'s rows. Accordingly, while Equation (6) is left multiplied by $A$, it is transferred to

$$P = AI = AMW, \tag{7}$$

where $P$ is the coding matrix of RGB image, Depth image, and 2DM image for $I$. From Equation (7), we can find

$$W = KP, \tag{8}$$

where the unknown matrix $K$ can be obtained by $K = (AM)^{-1}$. In Equation (8), $W$, $P$ are the joint coding of the three-dimensional point and its two-dimensional point obtained by the auto-encoder. When the RGB-D camera is turned on, the camera can automatically generate multiple joint codes $P$ from the RGB, Depth image, and the corresponding 3D position $W$ automatically. The VJTR method proposed in this paper is to obtain $K$ from each of the variables $P$ and $W$. Our objective function is expressed as

$$K = \arg_{\forall(P,W)} \min \|KP - W\|. \tag{9}$$

From Equation (9), we can get the estimated value of the true three-dimensional position of the $K_{real}$ in the world coordinate system.

As shown in Figure 3, we use the ResNet-152 network to simultaneously estimate the three-dimensional position. We will jointly encode the RGB' image and 2DM image, which are segmented by the YOLACT++ network, and the Depth image collected by the RGB-D camera into the ResNet-152 network, and finally output the 3D estimated position of the object. The ResNet-152 network is similar to other residual networks. The convolution module has 5 modules from conv1, conv2_x to conv5_x. In conv1, we use the convolution kernel of $7 \times 7$ to convolve the input image of $224 \times 224 \times 3$. This layer has 64 channels with a step size of 2 to obtain a $112 \times 112 \times 64$ feature map. Then, we input the feature map to the maximum pooling layer with a step size of 2, and a convolution kernel of $3 \times 3$ to

perform feature extraction. Finally, we obtain a $56 \times 56 \times 64$ feature map, and input it into the residual layer. The specific structure of the residual layer is shown in Table 1.

**Table 1.** Detailed parameters of hourglass structure.

| Network Layer | Kernel Parameter | Repeat | Output |
|---|---|---|---|
| Input | | $56 \times 56 \times 64$ | |
| Block1 | $\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix}$ | 3 | $56 \times 56 \times 64$ |
| Block2 | $\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix}$ | 8 | $28 \times 28 \times 128$ |
| Block3 | $\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix}$ | 36 | $14 \times 14 \times 256$ |
| Block4 | $\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix}$ | 3 | $7 \times 7 \times 512$ |

### *2.4. Reconstruction Error Correction*

In the process of obtaining 3D reconstruction point set $P$ which uses the above method, some outliers will be generated. Therefore, we propose an outlier adjustment method based on cluster center distance constrained (BCC-Drop) to improve the reconstruction accuracy.

Assume that $P$ contains m points: $p_1, \ldots p_i, \ldots p_m$, where $p_i \in R^S$, its initial cluster center point $p_0$ is derived from $p_0 = \sum_{i=1}^{m} p_i / m$. For each point $p_i$, we set a corresponding new set of $N$, the set $n_i = \|p_i - p_0\| (1 \leq i \leq m)$ is composed of $n_i$. Then, according to the literature [43], the local density and distance $\delta_i$ for each $n_i$ are given as:

$$\rho_i = \sum_{j=1, j \neq i}^{m} N(d_{ij} - d_c), \tag{10}$$

$$\delta_i = \min_{j:\rho_j > \rho_i}(d_{ij}). \tag{11}$$

In Equation (10), $d_{ij}$ is the distance between the terms $n_i$ and $n_j$. If $n < 0$, the function $N(n) = 1$, otherwise $N(n) = 0$. $d_c$ is a cutoff distance, which is suggested to be the values that ensure the average number of neighbors is around 1 to 2% of the total number of points in the dataset [44]. In Equation (11), $\delta_i$ is measured by the minimum distance between the item $x_i$ and any other item with higher density. Then, according to the definition of BCC-DROP [43], the items which have the max values of $\delta$ are the cluster center points of good items, and the items with low values of $\rho$ are outliers. We divide different points into the same cluster by $d_c$, and then determine whether the point is a cluster center point or an outlier by calculating the distance from the point with higher density. After finding the better set $N_{in} = N - N_{out}$, where $N_{out}$ is the abnormal item in $N$, we can obtain the corresponding good point set $P_{in}$. After finding $P_{in}$, we use

$$\overline{x}_i = (x_i - x_{\min}) / (x_{\max} - x_{\min}), \tag{12}$$

$$\overline{y}_i = (y_i - y_{\min}) / (y_{\max} - y_{\min}), \tag{13}$$

$$\overline{z}_i = (z_i - z_{\min}) / (z_{\max} - z_{\min}), \tag{14}$$

to adjust each point of $p_i = (x_i, y_i, z_i) \in P_{in}$. In Equations (12)–(14), the range of $x_i, y_i, z_i$ is (0, 1], where $x_{\max} = \forall_i \max(x_i)$, $x_{\min} = \forall_i \min(x_i)$, $y_{\max} = \forall_i \max(y_i)$, $y_{\min} = \forall_i \min(y_i)$, $z_{\max} = \forall_i \max(z_i)$, and $z_{\min} = \forall_i \min(z_i)$ are the coordinates of the maximum and minimum values of $P$. Finally, we obtain the point set $\overline{P}$ $\left(\overline{p}_i = (\overline{x}_i, \overline{y}_i, \overline{z}_i) \in \overline{P}\right)$ which is normalized. The point set $\overline{P}$ is the real-time 3D reconstruction model of the object.

### *2.5. Method Summary*

The whole process of the algorithm proposed in this article is summarized in Algorithm 1. Firstly, we adjust the image size and input them into the neural network in batches in the training phase of ResNet-152. Secondly, we use the gradient function to minimize the loss function to train the neural network. Finally, the model with the highest target detection accuracy is selected according to the validation set. In the same way, we used a similar method in the training stage of the YOLACT++ neural network.

**Algorithm 1.** The process of the real-time 3D reconstruction method

The RGB and Depth images of the scene are collected by the monocular camera.
Input RGB to YOLACT++ to obtain the segmented RGB', 2DM.
Input RGB', 2DM, Depth to VJTR to get $P$.
Set $p_0 \leftarrow \sum_{i=1}^{m} p_i/m$, **then** let $\delta_{\max} \leftarrow -1.0$ and $s \leftarrow -1$.
**for** Each $p_i$
  Use Formula (10) to calculate the $\rho_i$.
  **for** $i$ = 1 to m step **do**
   Use Formula (11) to calculate the $\delta_i$.
   **if** ($\delta_i > \delta_{\max}$) **then**
   $\delta_{\max} \leftarrow \delta_i$.
   $s \leftarrow i$.
   **end if**
  **end for**
**end for**
**for** $i$ = 1 to m step **do**
  **if**($\|p_i - p_0\| < \lambda \|p_s - p_0\|$) **then**
   Add $p_i$ to $P_{in}$.
  **else**
   Add $p_i$ to $P_{out}$.
  **end if**
**end for**
**for** Each $p_i$
Use Formulas (12)–(14) to normalize its coordinate values.
Add $p_i$ to $\overline{P}$.
**end for**
**return** $\overline{P}$.

## 3. Experiments

In this section, we first introduce the experimental environment and visual information extraction results, and compare the segmentation performance of different instance segmentation methods on the COCO dataset. Then, we describe the implementation of VJTR and compare the experimental results of several different types of input combinations and different types of CNN 3D position estimation. After that, we compare the model obtained by using the neural network for 3D position estimation with the model obtained by using the BCC-Drop method. NN represents the experimental results which use the neural network. BCC-Drop-NN represents the experimental results which use the neural network first to obtain the three-dimensional scene point cloud, and then use the BCC-Drop method to remove outliers. Finally, we detail the time cost of each step for the proposed method. In the experimental results section, we add a comparison between our method and other main methods on the YCB-Video dataset.

### *3.1. Experimental Setting*

This experiment is based on Windows 64-bit platform. We use the development language C++ as it has the advantages of high efficiency, simplicity, complete third-party libraries, and portability. The third-party development libraries used are OpenCV and OpenGL. OpenCV is used to process image data, and OpenGL is used for real-time re-

construction and visualization. The handheld RGB-D camera dataset is collected by the kinect2.0 camera. The RGB-D camera is connected to a 3.2 GHz i7-8700 CPU, 16.0 G RAM, and NVidia GTX 1660 graphics computer. We use the standard dataset COCO to train, test and validate the YOCACT++ model. For the VJTR method, we use 11,356 sets of data collected by a handheld RGB-D camera as the training set, and use the standard data-set YCB-Video to test and verify the effectiveness of our method.

### *3.2. Implementation and Results of Visual Information Segmentation Extraction*

We use YOLACT++ network to extract objects from RGB images. Before inputting the images to the YOLACT++ network, we adjust them to images with a resolution of $550 \times 550$. We use different types of instance segmentation networks to segment and extract visual information on the COCO dataset. As shown in Table 2, the Mask Scoring R-CNN network has the best segmentation effect, but its average running time is as high as 116.3 ms. The extraction of visual information will greatly reduce the speed of real-time reconstruction in this way. Compared with other mainstream segmentation networks, the YOLACT++ network can reach 33.5 FPS, which meets the needs of real-time reconstruction. The average precision is 34.1. It can be seen that under the premise of ensuring real-time segmentation, YOLACT++ can segment objects more accurately and reduce reconstruction errors.

**Table 2.** Segmentation experiment results of different types of instance segmentation networks on the COCO dataset.

| Number | Method | Average Precision | FPS | Time (ms) |
|---|---|---|---|---|
| 1 | PA-Net [45] | 36.6 | 4.7 | 212.8 |
| 2 | RetinaMask [46] | 34.7 | 6.0 | 166.7 |
| 3 | FCIS [47] | 29.5 | 6.6 | 151.5 |
| 4 | Mask R-CNN [48] | 35.7 | 8.6 | 116.3 |
| 5 | Mask Scoring R-CNN [49] | 38.3 | 8.6 | 116.3 |
| 6 | YOLACT++ | 34.1 | 33.5 | 29.9 |

### *3.3. VJTR Method Realization and Results*

We input the image RGB', 2DM, and Depth which are segmented by YOLACT++ to different types of CNN networks. The experimental results of deep recovery are listed in Table 3, in which we select VGG-16, InceptionNet-V3 and ResNet-152 as different types of input combinations. In the input column of Table 3, C, D, and M, respectively represent RGB' image, Depth, and 2DM mask image. As the input requires the depth map corresponding to the RGB image, we use the data collected by our handheld RGB-D camera as the training set, and use the public dataset YCB-Video for verification.

From Table 3, we can see that ResNet network used as dimension reduction for $C$, $D$, $M$ performs better than those of VGG and InceptionNet on $AE$ and $ME$. We believe that the smaller the average error and maximum error, the better the performance. We can find that $D + M$ (2DM and Depth) is the basic effective channel used for depth recovery, while the channel $C$ (RGB' image) has little help in reducing the depth recovery error. It only provides two-dimensional image information that requires three-dimensional reconstruction. Secondly, we can see that the maximum reconstruction error value of $D$ is much greater than the maximum reconstruction error value of $D + M$. Due to the measurement error caused by backlight, objects outside the reflection range, or distant objects, and multiple infrared reflections in the environment, there are a lot of noise and pits in the Depth. A single pixel may be located in the noise range or recessed, resulting in a lot of inaccurate depth information. $M$ (2DM mask image) can provide $5 \times 5$ average information which can reduce the probability of maximum errors in visual reconstruction points. As shown in Figure 4, Figure 4a is the distance of the object's red line relative to the real ground, and Figure 4b is the point cloud model obtained by deep fusion of the RGB image collected by

RGB-D and the Depth image, Figure 4c is the model obtained by the VJTR method. We use OpenGL to visualize the point cloud information generated in this article, and at the same time obtain the color information of the point in the RGB image through the pixel value. For the teacup handle, due to its irregular shape and small volume, the depth value is very easy to take an abnormal value or a null value. We can clearly see that the proposed VJTR method effectively reduces the depth error in the reconstruction of the contour edge of the object. In addition, we can see that it is time-consuming to restore the depth values of all points of the object from the image, so we only reconstruct the selected $k$ points in the actual process. Suppose the number of object points obtained through the YOLACT++ network is $w$, where $k = w/5$.

**Table 3.** The results of joint coding of different input combinations and different types of CNN depth recovery on the YCB-Video dataset.

| Number | Neural Networks Category | Enter | Average Error (m) | Maximum Error (m) | Parameter Size (MB) | Time (s) |
|---|---|---|---|---|---|---|
| 1 | VGG-16 | C + D + M | 0.041 | 0.098 | 46.20 | 0.159 |
| 2 | | C + D | 0.056 | 0.113 | 39.96 | 0.176 |
| 3 | | D + M | 0.040 | 0.079 | 21.71 | 0.169 |
| 4 | | D | 0.030 | 0.101 | 20.04 | 0.159 |
| 5 | InceptionNet-V3 | C + D + M | 0.049 | 0.094 | 27.77 | 0.151 |
| 6 | | C + D | 0.051 | 0.104 | 22.63 | 0.13 |
| 7 | | D + M | 0.026 | 0.063 | 21.63 | 0.129 |
| 8 | | D | 0.031 | 0.096 | 20.04 | 0.125 |
| 9 | ResNet-152 | C + D + M | 0.037 | 0.085 | 13.47 | 0.121 |
| 10 | | C + D | 0.044 | 0.068 | 12.61 | 0.115 |
| 11 | | D + M | 0.017 | 0.052 | 8.79 | 0.115 |
| 12 | | D | 0.041 | 0.070 | 6.52 | 0.096 |

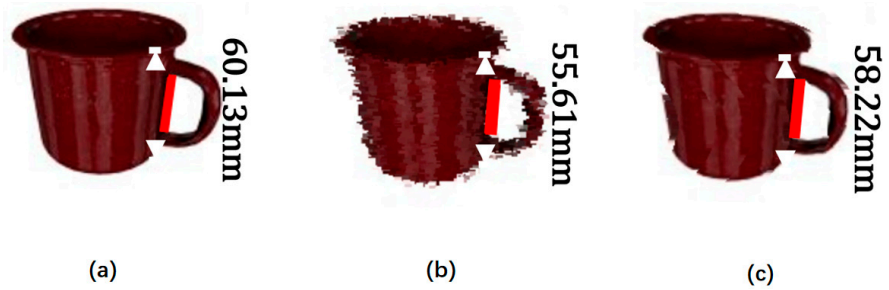


**Figure 4.** Comparison of teacup reconstruction models on YCB-Video dataset. (**a**) RGB image; (**b**) NN results; and (**c**) BCC-Drop results.

### *3.4. BCC-Drop Strategy Implementation and Results*

Figure 5b,c show the segmentation results and selected partial contour points of m objects obtained by YOLACT++. The w and k values of bottle, blue barrel, brown box, blue cans, and red bowl from top to bottom are 684, 440, 2680, 260, and 468, and 137, 88, 536, 52, and 94, respectively. When we determine the points that need to be reconstructed, we use ResNet-152 neural network to estimate them.

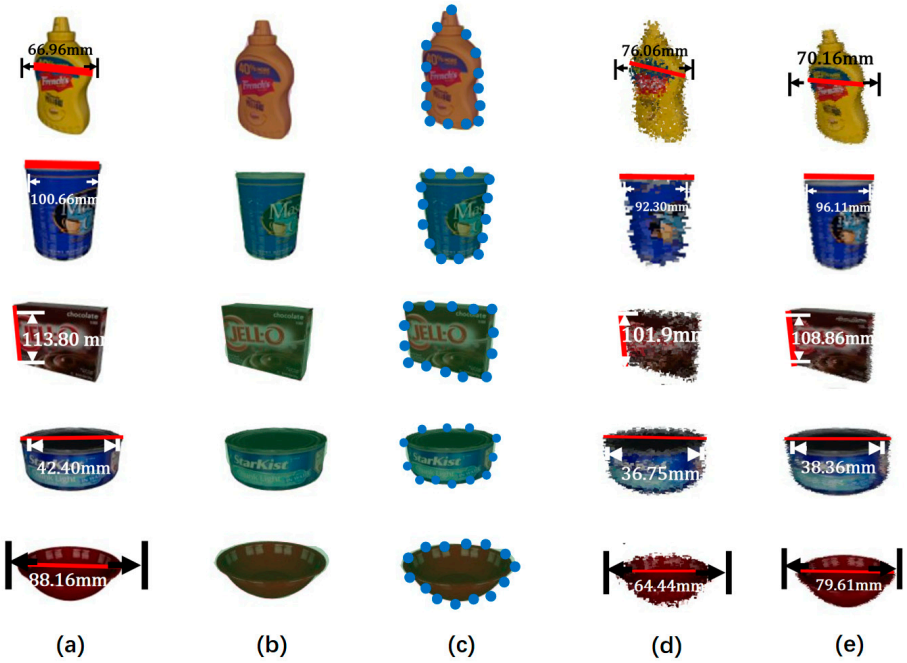


**Figure 5.** Extract and reconstruct samples of some objects in the experiment. (**a**) RGB image; (**b**) YOLACT++ segmentation results; (**c**) Schematic diagram of selected contour points; (**d**) NN result; and (**e**) BCC-Drop results.

Figure 5d,e show the object reconstruction obtained by a single neural network (NN) method and the BCC-Drop strategy proposed for Figure 5a, where the red line in Figure 5b represents the true ground distance between two contour points. The corresponding red lines in Figure 5d,e indicate that the reconstruction distance of NN and BCC-Drop-NN, respectively. Assuming that the length of the red line ground true distance is $L_j^{GT}$, the red line reconstruction length NN and BCC-Drop-NN are, respectively $L_j^{NN}$ and $L_j^{DP}$ where $j(0 \leq j \leq J)$ is the total number of red line segments for different positions of the object, to be used to compare experimental results. We use

$$Err_{NN} = \frac{\sum_{j=1}^{J} e_j^{NN}}{J}, \text{ where } err_{NN} = \frac{\|L_j^{NN} - L_j^{GN}\|}{L_j^{GN}}, \tag{15}$$

$$Err_{DP} = \frac{\sum_{j=1}^{J} e_j^{DP}}{J}, \text{ where } err_{DP} = \frac{\|L_j^{DP} - L_j^{GN}\|}{L_j^{GN}}, \tag{16}$$

to calculate the reconstruction error of NN and BCC-Drop-NN, respectively.

The values in the accuracy column of Table 4 represent the absolute and relative errors of the object given in Figure 5a, where $Err_{NN}$ and $Err_{DP}$ the relative errors. For each object, we calculate at least twelve lines and their distances, that is $J \geq 12$. It can be seen from Table 4 that the absolute error of reconstruction obtained by BCC-Drop-NN is significantly smaller than that of NN reconstruction. Although similar reconstruction effects are obtained by the BCC-Drop-NN method and NN, we can clearly see that the relative error of the bottle is about 9.16% and 12.29%, respectively, and the blue barrel is about 5.65% and 9.71%, respectively or the red bowl, its values are about 6.53% and 10.75%, respectively. From this experimental result, the relative error obtained by our proposed BCC-Drop-NN method is smaller than that of the NN method. In general, the average total error value of the BCC-Drop-NN method is about 3.7% less than the average total error value of the NN method.

**Table 4.** The reconstruction accuracy and time cost on the YCB-Video dataset.

| Object | Point | | Precision | | | | Time (ms) | | |
|---|---|---|---|---|---|---|---|---|---|
| | $m$ | $k$ | $\frac{\sum_j (L_j^{NN} - L_j^{GT})}{J}$ | $\frac{\sum_j (L_j^{DP} - L_j^{GT})}{J}$ | $Err_{NN}$ (%) | $Err_{DP}$ (%) | $T_{\text{YOLACT++}}$ | $T_{\text{VJTR}}$ | $T_{\text{BCC-Drop}}$ |
| Bottle | 684 | 137 | 8.62 mm | 6.43 mm | 12.29 | 9.16 | 30.12 | 2.32 | 1.14 |
| Blue barrel | 440 | 88 | 9.77 mm | 5.69 mm | 9.71 | 5.65 | 30.01 | 1.35 | 0.44 |
| Brown box | 2680 | 536 | 11.13 mm | 6.94 mm | 9.78 | 6.10 | 30.33 | 5.26 | 2.21 |
| Blue cans | 260 | 52 | 3.58 mm | 2.04 mm | 8.44 | 4.81 | 30.06 | 1.22 | 0.23 |
| Red bowl | 468 | 94 | 9.48 mm | 5.76 mm | 10.75 | 6.53 | 30.03 | 1.48 | 0.51 |
| Mean | / | / | 8.52 mm | 5.37 mm | 10.19 | 6.45 | 30.11 | 2.32 | 0.90 |

The three columns of time cost in Table 4 detail the time cost of each stage. It is worth noting that for a given RGB image, the time overhead of YOLACT++ is about 30.11 ms, which has nothing to do with the number of objects in the scene. This is as YOLACT++ always calculates the probability of whether each pixel in the image is a boundary point of an object. For the depth estimation and anomaly elimination steps, the average cost time of these two steps is about 2.32 ms and 0.90 ms. The time cost of these two parts is closely related to the number of scene reconstruction points, so we only select some points for reconstruction. The reconstruction time cost of this method is about 0.033 s, 30 or 31 frames per second (FPS).

### *3.5. Experimental Results*

Figure 6 shows the reconstruction effects of different methods on the same public dataset YCB-Video. We can see that, although the method in Figure 6b reconstructs most of the elements of the scene, many of these elements need not to be reconstructed. This method not only wastes a lot of time, but also has low reconstruction accuracy. Although the method in Figure 6c has a fast reconstruction speed, its reconstruction depth value error is relatively large, which makes the object broken after reconstruction. It fails to complete the reconstruction. In addition, in the scene in Figure 7a, we can see that these objects are placed closely together, there is no distance between them, and there is obvious overlap and occlusion. For example, a blue box is placed on a white box, and the white box is blocked by a dark cup and a white cup. Even so, it can be seen from Figure 7d–f that the reconstruction points of these objects are correctly positioned at their respective positions. Take the white box as an example. In the views shown in Figure 7a,b, a white cup and a dark gray cup are placed to block, and there is even a blue box stacked with it. Two objects can still be distinguished well in the reconstruction view of Figure 7d,f. It shows that, even if objects are placed closely or even stacked in the RGB-D image, their 3D reconstruction points are distributed correctly in space, and their spatial relationship in the real 3D space is also well reflected.

In addition, it can be seen from the reconstruction results in Table 4 and the real 3D reconstruction results in Figure 7 that the proposed method can complete real-time 3D reconstruction of monocular vision. With an average absolute reconstruction error of 5.37 mm and a relative error of 6.45%, this method can achieve accurate reconstruction of objects in a virtual three-dimensional space, even if these objects are closely placed in the scene in a stacked and overlapping manner. It lists the comparison of time cost and accuracy between our real-time 3D reconstruction method and other real-time 3D reconstruction methods in Table 5. It is not difficult to see that the real-time reconstruction method based on monocular vision proposed by us not only requires a single camera to complete real-time reconstruction, but also greatly improves the reconstruction speed and reconstruction accuracy.

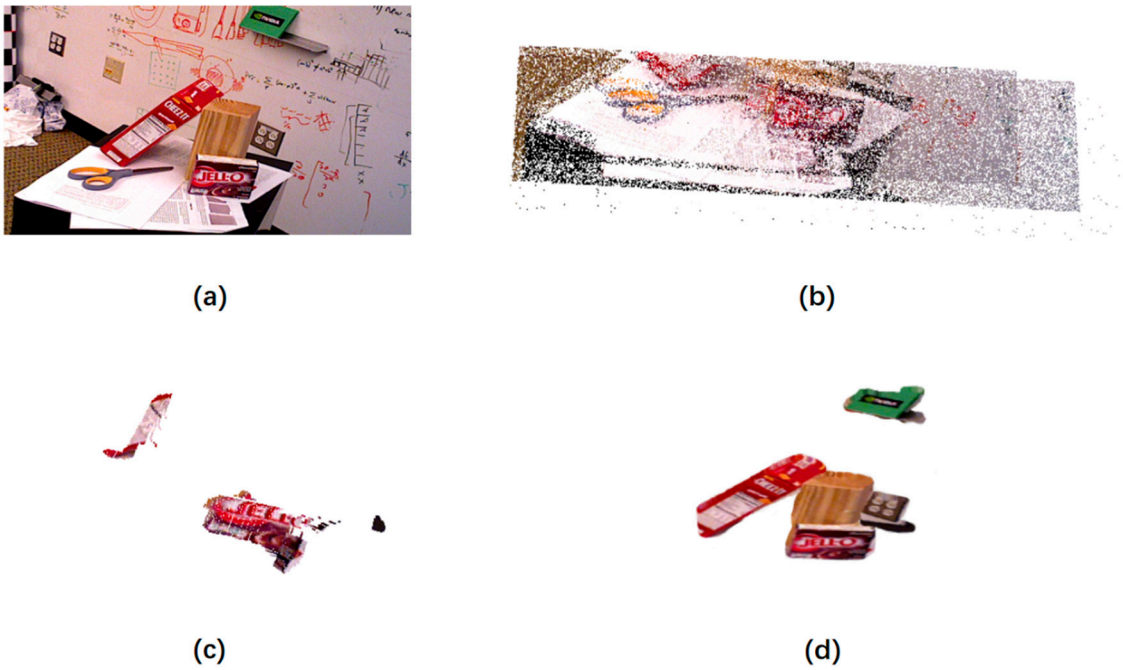


**Figure 6.** The experimental results of our method and other main methods on the YCB-Video dataset. (**a**) YCB-Video data RGB image; (**b**) ORB reconstruction results; (**c**) EKF-SLAM reconstruction results; and (**d**) OURS reconstruction results.

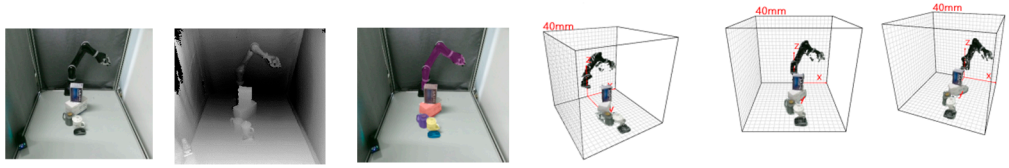


**Figure 7.** Real-time reconstruction results of monocular cameras in complex scenes. (**a**) RGB image; (**b**) Depth image; (**c**) Object extraction results obtained by YOLACT++; and (**d**–**f**) Real-time 3D reconstruction results from different views.

**Table 5.** The performance of our method and other main methods on the YCB-Video dataset.

| Algorithm | Err (%) | Running Time (ms) | Detection Speed (fps) |
|---|---|---|---|
| EKF-SLAM | 18.41 | 80.11 | 12 |
| ORB | 10.34 | 251.32 | 3 |
| OURS | 6.71 | 33.64 | 29 |

## 4. Conclusions

In this work, we propose a real-time reconstruction method based on a monocular camera. The YOLACT++ network is used to extract the objects that need to be reconstructed in the RGB-D image from the clustering environment. We use ResNet-152 neural network to propose a three-dimensional position estimation method (VJTR) for joint coding of visual information. This method combines three stages of depth recovery, depth optimization, and depth fusion to quickly and accurately obtain the three-dimensional point value of the object extracted by the YOLACT++ network from the RGB-D image. A method is proposed to limit outlier adjustment based on the distance of cluster centers, namely the BCC-Drop method. It can remove the partial separation group value and reduce the three-dimensional reconstruction error. We verified the effectiveness of this method on the actual complex stacked 3D scene and the public dataset YCB-Video. The experimental results show that the reconstruction speed of this method is 30 FPS, and the relative error

of reconstruction is 9%. Compared with other mainstream real-time 3D reconstruction methods, this method has stronger real-time performance on ordinary PCs and better real-time reconstruction effects in a real cluster environment. In addition, our method can automatically organize the reconstructed objects in the three-dimensional space, and extract and label the reconstructed objects through the YOLACT++ network in a clustering environment. Using these tags obtained by the YOLACT++ network, the reconstructed points of the object can also be automatically and synchronously marked in the three-dimensional space. It should be noted that our method does not optimize the phase of RGB-D camera acquisition of two-dimensional images, which will be our future work.

**Author Contributions:** Conceptualization, W.X.; methodology, B.Q. and X.Y.; software, Y.S.; validation, S.Z.; formal analysis, Q.J.; writing—original draft preparation, L.C.; writing—review and editing, M.Y.; visualization, Q.J. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was partially supported by the National Natural Science Foundation of China (61762025), the Guangxi Key Research and Development Program (AB18126053, AB18126063, AD18281002, AD19110137), the Natural Science Foundation of Guangxi of China (2019GXNSFDA185007, 2019GXNSFDA185006), Guangxi Key Science and Technology Planning Project (AA18118031, AA18242028), Guilin Science and Technology Development Program (20190211-17). The Innovation Project of GUET Graduate Education (2020YCXS052), and the Guangxi Colleges and Universities Key Laboratory of Intelligent Processing of Computer Image and Graphics (No. GIIP201603).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** This study analyzed publicly available data sets. These data can be found here: [https://rse-lab.cs.washington.edu/projects/posecnn/] (accessed on 28 August 2021).

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Yu, C.; Ji, F.; Jing, X.; Liu, M. Dynamic Granularity Matrix Space Based Adaptive Edge Detection Method for Structured Light Stripes. *Math. Probl. Eng.* **2019**, *2019*, 1959671. [CrossRef]
2. Feri, L.E.; Ahn, J.; Lutfillohonov, S.; Kwon, J. A Three-Dimensional Microstructure Reconstruction Framework for Permeable Pavement Analysis Based on 3D-IWGAN with Enhanced Gradient Penalty. *Sensors* **2021**, *21*, 3603. [CrossRef]
3. Li, H.; Wang, R. Method of Real-Time Wellbore Surface Reconstruction Based on Spiral Contour. *Energies* **2021**, *14*, 291. [CrossRef]
4. Storms, W.; Shockley, J.; Raquet, J. Magnetic field navigation in an indoor environment. In Proceedings of the 2010 Ubiquitous Positioning Indoor Navigation and Location Based Service, Kirkkonummi, Finland, 14–15 October 2010; pp. 1–10.
5. Slavcheva, M.; Baust, M.; Ilic, S. Variational Level Set Evolution for Non-Rigid 3D Reconstruction from a Single Depth Camera. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *43*, 2838–2850. [CrossRef]
6. Fei, C.; Ma, Y.; Jiang, S.; Liu, J.; Sun, B.; Li, Y.; Gu, Y.; Zhao, X.; Fang, J. Real-Time Dynamic 3D Shape Reconstruction with SWIR InGaAs Camera. *Sensors* **2020**, *20*, 521. [CrossRef] [PubMed]
7. Wen, Q.; Xu, F.; Yong, J. Real-Time 3D Eye Performance Reconstruction for RGBD Cameras. *IEEE Trans. Vis. Comput. Graph.* **2017**, *23*, 2586–2598. [CrossRef]
8. Gu, Z.; Chen, J.; Wu, C. Three-Dimensional Reconstruction of Welding Pool Surface by Binocular Vision. *Chin. J. Mech. Eng.* **2021**, *34*, 47. [CrossRef]
9. Yuan, Z.; Li, Y.; Tang, S.; Li, M.; Guo, R.; Wang, W. A survey on indoor 3D modeling and applications via RGB-D devices. *Front. Inf. Technol. Electron. Eng.* **2021**, *22*, 815–826. [CrossRef]
10. Lu, F.; Peng, H.; Wu, H.; Yang, J.; Yang, X.; Cao, R.; Zhang, L.; Yang, R.; Zhou, B. InstanceFusion: Real-time Instance-level 3D Reconstruction Using a Single RGBD Camera. *Comput. Graph. Forum* **2020**, *39*, 433–445. [CrossRef]
11. Henry, P.; Krainin, M.; Herbst, E.; Ren, X.; Fox, D. RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. *Int. J. Robot. Res.* **2012**, *31*, 647–663. [CrossRef]
12. Vogiatzis, G.; Hernández, C. Video-based, real-time multi-view stereo. *Image Vis. Comput.* **2011**, *29*, 434–441. [CrossRef]
13. Engel, J.; Koltun, V.; Cremers, D. Direct Sparse Odometry. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 611–625. [CrossRef] [PubMed]
14. Stumberg, L.V.; Usenko, V.; Cremers, D. Direct Sparse Visual-Inertial Odometry Using Dynamic Marginalization. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 2510–2517.

15. Furukawa, Y.; Ponce, J. Accurate, Dense, and Robust Multiview Stereopsis. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *32*, 1362–1376. [CrossRef] [PubMed]
16. Jancosek, M.; Pajdla, T. Multi-view reconstruction preserving weakly-supported surfaces. In Proceedings of the CVPR 2011, Washington, DC, USA, 20–25 June 2011; pp. 3121–3128.
17. Newcombe, R.A.; Lovegrove, S.J.; Davison, A.J. DTAM: Dense tracking and mapping in real-time. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 2320–2327.
18. Wu, Z.; Wu, X.; Zhang, X.; Wang, S.; Ju, L. Semantic stereo matching with pyramid cost volumes. In Proceedings of the Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27–28 October 2019; pp. 7484–7493.
19. Gu, X.; Fan, Z.; Zhu, S.; Dai, Z.; Tan, F.; Tan, P. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 2495–2504.
20. Yang, Z.; Gao, F.; Shen, S. Real-time monocular dense mapping on aerial robots using visual-inertial fusion. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 4552–4559.
21. Schöps, T.; Sattler, T.; Häne, C.; Pollefeys, M. Large-scale outdoor 3D reconstruction on a mobile device. *Comput. Vis. Image Underst.* **2017**, *157*, 151–166. [CrossRef]
22. Azhar, N.; Saad, W.H.M.; Manap, N.A.; Saad, N.M.; Syafeeza, A.R. Silhouette-based approach of 3D image reconstruction for automated image acquisition using robotic arm. *IOP Conf. Ser. Mater. Sci. Eng.* **2017**, *210*, 012049. [CrossRef]
23. Bo, Z.-H.; Zhang, H.; Yong, J.-H.; Gao, H.; Xu, F. DenseAttentionSeg: Segment hands from interacted objects using depth input. *Appl. Soft Comput.* **2020**, *92*, 106297. [CrossRef]
24. Tong, J.; Zhou, J.; Liu, L.; Pan, Z.; Yan, H. Scanning 3D Full Human Bodies Using Kinects. *IEEE Trans. Vis. Comput. Graph.* **2012**, *18*, 643–650. [CrossRef] [PubMed]
25. Garcia-Fidalgo, E.; Ortiz, A. *Methods for Appearance-Based Loop Closure Detection: Applications to Topological Mapping and Image Mosaicking*; Springer: New York, NY, USA, 2018; Volume 122.
26. Maimone, A.; Fuchs, H. Encumbrance-free telepresence system with real-time 3D capture and display using commodity depth cameras. In Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality, Basel, Switzerland, 26–29 October 2011; pp. 137–146.
27. Alexiadis, D.S.; Zarpalas, D.; Daras, P. Real-Time, Full 3-D Reconstruction of Moving Foreground Objects from Multiple Consumer Depth Cameras. *IEEE Trans. Multimed.* **2013**, *15*, 339–358. [CrossRef]
28. Liu, S.-L.; Guo, H.-X.; Pan, H.; Wang, P.-S.; Tong, X.; Liu, Y. Deep Implicit Moving Least-Squares Functions for 3D Reconstruction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 9–16 November 2021; pp. 1788–1797.
29. Alexiadis, A. Deep multiphysics: Coupling discrete multiphysics with machine learning to attain self-learning in-silico models replicating human physiology. *Artif. Intell. Med.* **2019**, *98*, 27–34. [CrossRef] [PubMed]
30. Ceron, J.C.A.; Chang, L.; Ochoa-Ruiz, G.; Ali, S. Assessing YOLACT++ for real time and robust instance segmentation of medical instruments in endoscopic procedures. *arXiv* **2021**, arXiv:2103.15997. Available online: https://arxiv.org/abs/2103.15997 (accessed on 28 August 2021).
31. Wang, Z.; Xu, Y.; Yu, J.; Xu, G.; Fu, J.; Gu, T. Instance segmentation of point cloud captured by RGB-D sensor based on deep learning. *Int. J. Comput. Integr. Manuf.* **2021**, 1–14.
32. Bolya, D.; Zhou, C.; Xiao, F.; Lee, Y.J. Yolact++: Better real-time instance segmentation. *arXiv* **2019**, arXiv:1912.06218. [CrossRef]
33. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
34. Ngiam, J.; Khosla, A.; Kim, M.; Nam, J.; Lee, H.; Ng, A.Y. Multimodal deep learning. In Proceedings of the ICML, Bellevue, WD, USA, 28 June–2 July 2011.
35. Jo, H.; Kim, E. New Monte Carlo Localization Using Deep Initialization: A Three-Dimensional LiDAR and a Camera Fusion Approach. *IEEE Access* **2020**, *8*, 74485–74496. [CrossRef]
36. Zhao, C.; Sun, L.; Stolkin, R. A fully end-to-end deep learning approach for real-time simultaneous 3D reconstruction and material recognition. In Proceedings of the 2017 18th International Conference on Advanced Robotics (ICAR), Hong Kong, China, 10–12 July 2017; pp. 75–82.
37. Lombardi, M.; Savardi, M.; Signoroni, A. Cross-domain assessment of deep learning-based alignment solutions for real-time 3D reconstruction. *Comput. Graph.* **2021**, *99*, 54–69. [CrossRef]
38. Laidlow, T.; Czarnowski, J.; Leutenegger, S. DeepFusion: Real-time dense 3D reconstruction for monocular SLAM using single-view depth and gradient predictions. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 4068–4074.
39. Kim, M.; Jeng, G.-S.; Pelivanov, I.; O'Donnell, M. Deep-learning image reconstruction for real-time photoacoustic system. *IEEE Trans. Med Imaging* **2020**, *39*, 3379–3390. [CrossRef] [PubMed]
40. Visentini-Scarzanella, M.; Sugiura, T.; Kaneko, T.; Koto, S. Deep monocular 3D reconstruction for assisted navigation in bronchoscopy. *Int. J. Comput. Assist. Radiol. Surg.* **2017**, *12*, 1089–1099. [CrossRef] [PubMed]
41. Dai, J.; Qi, H.; Xiong, Y.; Li, Y.; Zhang, G.; Hu, H.; Wei, Y. Deformable convolutional networks. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 764–773.

42. Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the IEEE international conference on computer vision, Venice, Italy, 22–29 October 2017; pp. 2980–2988.
43. Rodriguez, A.; Laio, A. Clustering by fast search and find of density peaks. *Science* **2014**, *344*, 1492–1496. [CrossRef] [PubMed]
44. Meng, Y.; Zhuang, H. Self-Calibration of Camera-Equipped Robot Manipulators. *Int. J. Robot. Res.* **2001**, *20*, 909–921. [CrossRef]
45. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path aggregation network for instance segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8759–8768.
46. Jiang, M.; Fan, X.; Yan, H. Retinamask: A face mask detector. *arXiv* **2020**, arXiv:2005.03950. Available online: https://arxiv.org/abs/2005.03950 (accessed on 28 August 2021).
47. Li, Y.; Qi, H.; Dai, J.; Ji, X.; Wei, Y. Fully convolutional instance-aware semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2359–2367.
48. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2961–2969.
49. Huang, Z.; Huang, L.; Gong, Y.; Huang, C.; Wang, X. Mask scoring r-cnn. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 6409–6418.