第 58 卷 第 6 期/2021 年 3 月/激光与光电子学进展 研究论文

激光与光电子学进展 先进成像

基于图像处理的点云滤波算法

张建民[1]，陈富健[2]，龙佳乐[1*]

[1] 五邑大学智能制造学部，广东 江门 529020；

[2] 深圳大学电子与信息工程学院，广东 深圳 518060

摘要 在对点云进行三维重建中，不可避免地会产生各种噪声。现有方法中，在三维空间中通过分析和计算点云几何关系去除噪声点，存在计算复杂、耗时等缺点。提出了一种基于图像处理的点云滤波算法。首先对点云进行预处理，将点云映射到图像上，其次通过图像处理方法去除噪声区域，重新恢复成三维点云，最终得到不包含噪声的点云。详细分析了所提算法的实现过程，并通过实验结果验证了所提算法的有效性、实时性。

关键词 图像处理；点云；滤波；映射；实时性

中图分类号 TP391 **文献标志码** A **doi**：10.3788/LOP202158.0610015

Point Cloud Filtering Algorithm Based on Image Processing

Zhang Jianmin[1], Chen Fujian[2], Long Jiale[1*]

[1] *Faculty of Intelligent Manufacturing, Wuyi University, Jiangmen, Guangdong* 529020, *China*;

[2] *School of Electronic and Information Engineering, Shenzhen University, Shenzhen, Guangdong* 518060, *China*

Abstract Various types of noises are unavoidable during the three-dimensional reconstruction of a point cloud. In the existing methods, noise points are usually removed by analyzing and calculating the geometric relation of the point cloud in a three-dimensional space; this involves complicated and time-consuming calculations. This paper proposes a point cloud filtering algorithm based on image processing. First, the point cloud was preprocessed and mapped onto the image. Second, the noise area was removed from the image and restored to a three-dimensional point cloud by applying image processing. Thus, the point cloud without noise was obtained. This paper analyzes the implementation of the proposed algorithm in detail and verifies the effectiveness and real-time performance of the proposed algorithm based on experimental results.

Key words image processing; point cloud; filtering; mapping; real-time

OCIS codes 100.6890; 100.5088; 120.5050

1 引 言

基于结构光的三维测量[1-2]在古建筑、逆向工程、测量工程等领域中的应用越来越广泛，尽管扫描精度逐步提高，但不可避免地会从各种来源引入噪声，比如镜头失真、摄像机参数估计不准确等，在视觉测量过程中光线和算法误差同样会造成影响，因此对点云进行噪声处理是至关重要的。点云的噪声处理直接影响后续三维模型重建的质量、影响物体表面重建的准确性、影响 3D 模型制作结果的分析和使用等。

去除点云噪声、获取完整点云一直是各专家学者的研究重点。各专家学者针对三维测量中产生的噪声点，提出了各种去除方法[3-14]。在点云库

收稿日期：2020-08-12；**修回日期**：2020-08-24；**录用日期**：2020-09-03

基金项目：国家自然科学基金面上项目(61475120)、广东省自然科学基金面上项目(2020A1515010089)、广东省大学生科技创新培育专项资金(pdjh2020b0602)、校级高层次人才科研启动费(2017RC46)、江门市科技计划(2020JC01034、2020JC01028)

* **E-mail**：longjiale_528@126.com

(PCL)中[3]，集成了经典的点云滤波算法，如 k 近邻滤波算法[4]、直通滤波算法、半径滤波法、双边滤波算法[5-7]、体素网格滤波算法[8-9]等。k 近邻滤波算法能够去除离群噪声点中的散乱噪声点，但离群的块状噪声点由于面积较大无法去除；直通滤波算法是最简单的方法，能够快速去除一些离群噪声点，但滤除效果与点云的形状和位置有着很大的关系；双边滤波算法将图像处理中的方法应用到点云滤波上，但其中点云特征容易被平滑，而且不适用于滤除大块的噪声点和离群噪声点；体素网格滤波算法通过对点云建立一系列的体素网格，计算体素网格中的点云数量来确定是否滤除，但对输入数据较为敏感，并且对大块的噪声点滤除效果不佳。李仁忠等[10]提出一种基于方法库的点云去噪算法，即将 PCL 中经典的直通滤波、统计滤波、半径滤波、改进的双边滤波、体素栅格滤波结合起来对点云进行滤波。钟志鹏等[11]基于 OPTICS 聚类去除大尺度噪声和基于改进的双边滤波算法去除小尺度噪声。另外，随着深度学习的发展，Almonacid 等[12]将深度学习应用到点云滤波中，该方法的训练基于点云空间的体素化，其中体素包含了先验已知的噪声区域。经过训练后的卷积神经网络(CNN)能够识别点云中的噪声区域。

离群噪声点作为噪声点的一种类型，其分布没有规律，滤除也相对复杂，并且离群噪声点的存在严重影响着后续点云的处理。因此，针对离群噪声点，许多学者也提出了不同的方法予以解决。赵凯等[13]提出了改进的 DBSCAN 算法，该算法采用点云体素网格划分的方式加快对点云进行快速聚类，以分离出离群点。赵京东等[14]针对离群噪声点，通过改进基于曲面变化度的局部离群系数(SVLOF)方法来对噪声点进行识别。Nurunnabi 等[15]将鲁棒性和诊断性统计的基本思想结合在一起，提出了两种离群值检测和鲁棒显着性特征估计方法，该方法能够容忍更多的离群聚类值。杨雨薇等[16]针对离群噪声点，对点云构建拓扑关系，采用边界匹配和聚类的综合方法对点云进行滤波，取得了较好的效果。Arvanitis 等[17]提出了一种综合多方法的点云滤波方法，包括主成分分析(PCA)、k 最近邻、双边滤波，达到实时性的滤波效果。Ning 等[18]提出了一种基于两种几何特征约束的方法去除噪声点，即局部密度信息和局部拟合平面的偏差，但无法去除大块的噪声点。

基于结构光的三维测量设备获得的点云可能会存在大量的离群噪声点，而大多数点云滤波算法在三维空间中通过分析点云的几何关系来滤除噪声点，但由于点云数量十分庞大，直接在三维空间中对点云进行分析处理存在算法复杂、效率低下的缺点。本文提出一种基于图像处理的点云滤波算法，该算法不直接对三维空间中的点云进行分析处理，而是将点云映射到图像中，在二维平面中采用图像处理方法对点云进行处理，最后将点云重新映射回三维空间中，使得点云滤波更简单，达到实时性的滤波效果。

2 噪声特点分析

在物体三维重建过程中，不可避免会产生噪声。一是由于外界环境的影响，例如光照影响；二是由于测量设备的缺陷，例如镜头失真、镜头畸变、相机内部噪声等因素影响；三是由于算法的容忍噪声能力有限，超过算法的测量精度也会产生噪声；四是物体自身的特点，例如物体表面反光、物体表面具有不连续性、物体表面被遮挡等。产生的噪声点类型有散乱噪声点、块状噪声点、突刺噪声点等。图 1 为待测物体 1。在基于结构光的三维重构中，重建一个完整的物体需要对物体多个角度采集点云数据，且每次重建的点云只包含一个面，使用该方法产生的离群点主要为散乱的噪声点和块状的噪声，如图2、3

图 1 待测物体 1

Fig. 1 Object 1 to be measured

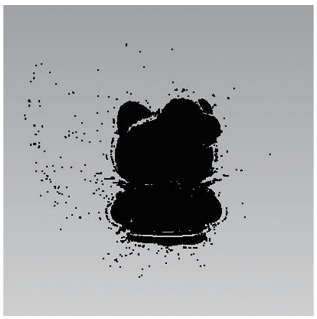

图 2 点云正视图

Fig. 2 Front view of point cloud

图 3 点云侧视图

Fig. 3 Side view of point cloud

所示。虽然存在大量的离群噪声点，但噪声点和物体点云之间存在着某种特定的关系，如图 3 所示，如果从侧面观看点云，物体点云是聚集的一大块区域，相反，噪声点是一些小块且位于物体点云附近。因此可以将点云视角旋转至侧面，找到点云侧视图中包含点数量最多的部分，滤除其他部分，即可完成对噪声点的滤除。

所提算法基于这个思路，首先将点云视角转为侧视，然后将点云映射到图像上，再通过图像处理方法确定物体点云对应位置，滤除其他点，从而得到不包含噪声的点云。所提算法根据图像的特点对点云进行分析计算，具有更高的效率，且能够达到实时性的滤波效果。

3 基本原理

3.1 基本框架

所提算法将点云映射到图像上，利用图像处理方法来滤除噪声点云部分，最后恢复出图像中对应的点云，具有简单、快速滤除噪声点的效果。算法主要分成 4 个步骤：点云预处理、将点云映射到图像上、图像处理、恢复点云。图 4 为所提算法过程示意图，大长方体表示物体点云，小长方体表示噪声点，总体过程如下。

1）先对点云进行预处理，将点云的坐标系旋转到以水平向右为 y 轴正方向，竖直向下为 x 轴正方向，垂直纸面向外为 z 轴正方向的坐标系，此时物体点云和噪声点云会被区分出来，如图 4(b)所示。

2）分别以图 4(b)中点云坐标沿 x 轴和 y 轴的最大值和最小值构建映射图像大小，将点云映射到 x-y 平面上，以区分噪声点云区域和物体点云区域，如图 4(c)所示。

3）对图 4(c)中的映射图像进行图像处理，滤除噪声点所对应区域，如图 4(e)所示。

4）恢复映射图像中对应的点云，最终得到不包含噪声的点云，如图 4(d)所示。

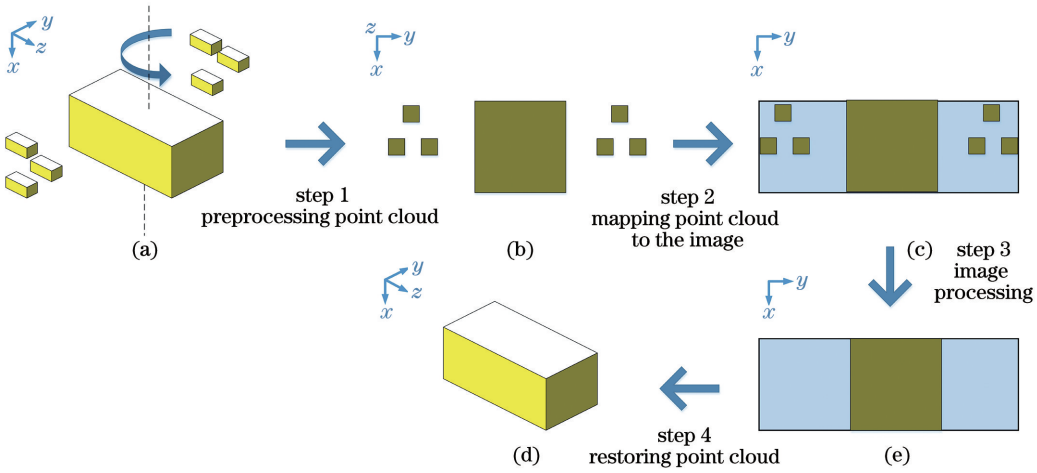

图 4 基于图像处理的点云滤波算法示意图

Fig. 4 Schematic of point cloud filtering algorithm based on image processing

具体地，这 4 个步骤中又分别包括几个小步骤，如下所示。图 5 为所提算法过程的具体实施流程图。点云预处理(step 1)：利用 k 最近邻算法对噪声点云进行滤除；计算点云坐标轴沿 x 方向和 y 方向的最小值，将点云沿 x 轴方向和 y 轴方向的坐标分别减去其对应的最小值；将点云坐标对应的浮点型数据转换为整型。

将点云映射到图像上(step 2)：对每一个点云数据进行编号，所得序号唯一；将点云的 x 轴坐标和 y 轴坐标映射到图像上；标记出每个图像坐标点 (u,v) 对应的点云序号和点云数量。

图像处理(step 3)：对映射图像进行闭运算操作；计算映射图像中的连通域并进行标记；计算每个连通域中对应的点云数量；选择面积最大的连通域或者选择包含点云数量最多的连通域予以保留，滤除其

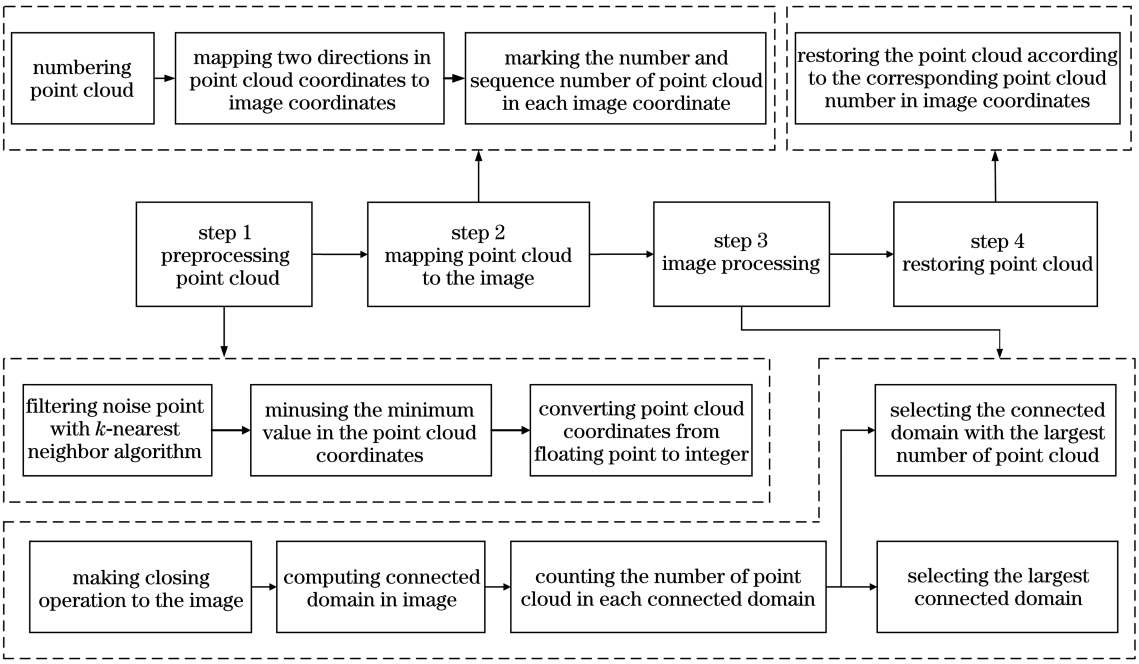

图 5 基于图像处理的点云滤波算法流程图

Fig. 5 Flow chart of point cloud filtering algorithm based on image processing

他区域。

恢复点云(step 4):经过 step 3 后得到了相应的图像坐标,根据图像坐标中对应的点云序号恢复点云,最终得到不包含噪声的点云。

3.2 图像预处理

由图 2、3 可知,物体点云附近存在非常多的散乱噪声点和块状噪声点,而散乱噪声点会使映射图像中物体点云和散乱点相连,对所提算法造成干扰。因此,可以先利用 k 最近邻算法滤除一些散乱的噪声点,通过对点云建立 kd 树后再利用 k 最近邻算法来对点云进行处理,能够大大加快算法速度,避免对点云滤波速度造成影响。

要将点云映射到图像上,首先要解决坐标系转换、数据类型不一致等问题。

一是坐标系转换问题。图像坐标系中存在坐标原点,而三维空间中的原点(0,0,0)在点云中几乎不存在,并且点云还可能存在负值,但图像坐标只有正值。将点云映射到图像上时,点云中每个点都必须对应到图像中每个点,因此要将点云坐标中最小值的点对应到图像坐标中的原点。设点云坐标为(x_i, y_i, z_i),转换后的点云坐标为(x_j, y_j, z_j),其中 $i=j$ 且 $i, j=1,2,3,\cdots,n$,则有

$$\begin{cases} x_j = x_i - \min(x_i) \\ y_j = y_i - \min(y_i) \\ z_j = z_i - \min(z_i) \end{cases} 。 \tag{1}$$

经过(1)式转换的点云中最小值的点为(0,0,0),并且不存在负值的情况,因此点云坐标最小值的点对应到图像坐标的原点(0,0),点云的每个点都能够与图像中的像素点对应起来。

二是数据类型不一致问题。由于点云坐标精度较高,通常为浮点型数据,而图像坐标的精度为整型,因此首先要将点云坐标从浮点型转换为整型。设点云浮点型坐标为(x_k, y_k, z_k),其中 $k=1,2,3,\cdots,n$,则有

$$\begin{cases} x_k = \mathrm{round}(x_j) \\ y_k = \mathrm{round}(y_j) \\ z_k = \mathrm{round}(z_j) \end{cases}, \tag{2}$$

式中:round(•)表示四舍五入。最终,以点云原点和坐标系中 x 轴和 y 轴建立的映射图像的长(H)和宽(W)分别为

$$\begin{cases} H = \max(y_k) - \min(y_k) \\ W = \max(x_k) - \min(x_k) \end{cases} 。 \tag{3}$$

如图 6 所示,最左边表示原始点云在三维空间中的拓扑结构,然后对原始点云中的每个点进行一一处理。其中,第一列为原始点云坐标,为浮点型数据。经过数据类型转换和坐标系转换后,得到第二列整型的数据。

3.3 将点云映射到图像上

将三维空间中的点云映射到二维空间的图像中时,空间的降维会导致出现三维空间中的多个点映射到二维空间图像上的一个点的情况,因此需要对点云中每一个点都进行编号。

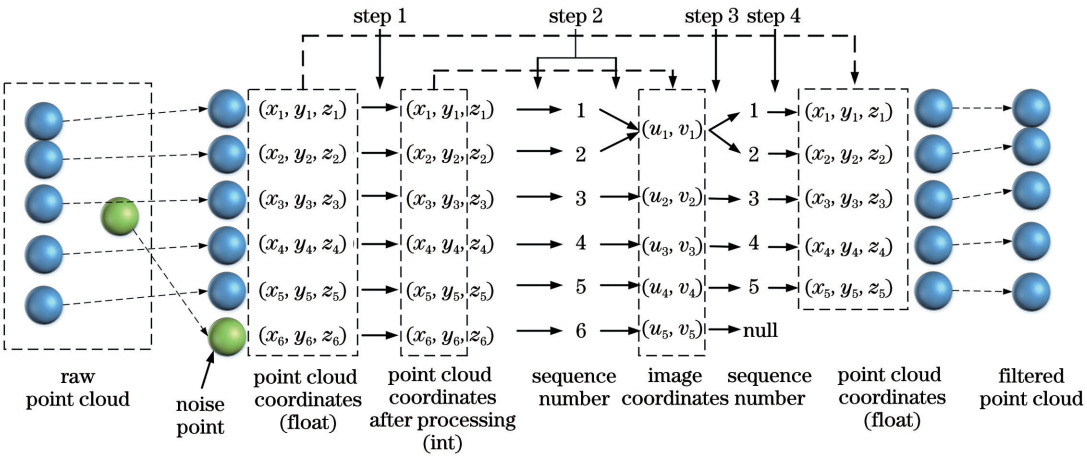

图 6 点云坐标、图像坐标和点云序号对应关系图

Fig. 6 Correspondence diagram of point cloud coordinates, image coordinates, and point cloud number

1) 对点云中的每个点从 1～n 进行编号，并且每个序号都是唯一的，每个序号唯一对应着点云中的每一个点。

2) 将点云坐标(x，y)映射到图像坐标(u，v)，(x，y)大小和(u，v)数值一致。

3) 同时标记图像坐标(u，v)对应的点云序号和对应的点云数量。

如图 6 所示，在 step 2 中对每个点都进行编号，得到第三列的序号 1～6，并将第二列中整型的点云坐标(x，y)映射到图像坐标(u，v)上，与此同时每个图像坐标都对应着点云的序号。

3.4 图像处理

将点云映射到图像上后，将图像中存在映射点云的图像坐标值设为 1，否则设为 0。因此映射图像为一张二值图像，基于二值图像进行图像处理操作更为高效快速。在图像中包含点云的两个信息：一是点云数量，二是点云序号。通过统计点云的数量从而确定物体点云和噪声点云的位置，根据图像坐标中对应点云序号恢复点云。

在点云映射到图像上时，物体点云对应的区域可能会存在一些断裂部分，导致当采用连通域算法计算时，物体点云对应的区域错误地被划分为不同的连通域。在此之前需先对映射图像进行闭运算，弥合映射图像中的一些断裂部分。

下一步计算映射图像中的连通域并进行标记。一般情况下，面积最大的连通域为物体点云对应的区域，其他一些小区域为噪声点对应的区域，但在特殊情况下，噪声点对应区域有可能大于物体点云对应的区域。一般选择映射图像中面积最大的连通域为物体点云对应的区域，以降低运算量，进一步提高算法的效率。通过统计每个连通域中的点云数量，选择包含点云数量最多的连通域为目标点云对应的区域，以能正确滤除噪声点。1)选择面积最大连通域。对每个点进行预处理后映射到图像上，结果如图 7(b)所示，点云中的每个点都对应着图像中的每个坐标点，并且图像坐标点对应着每个点云的序号。其中，在点云中紧密相邻的点会被映射到同一个图像坐标上，例如点 11 和点 15，但这并不影响点云的恢复，因为已经对点云中的每个点都进行了编号，并且序号具有唯一性。经过图像闭运算、计算图像的

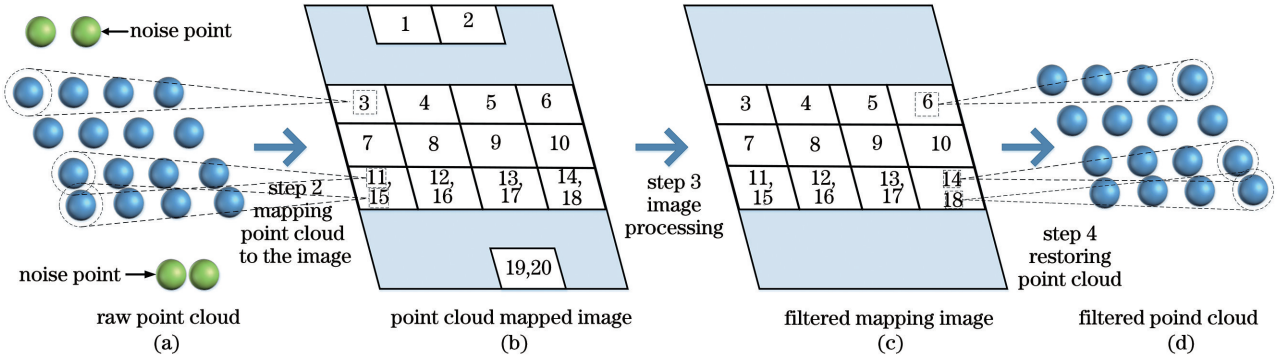

图 7 图像处理示意图(选择面积最大的连通域)

Fig. 7 Schematic of image processing (selecting the largest connected area)

连通域后，选择映射图像中面积最大的连通域，滤除其他部分，结果如图 7(c)所示，经过点云恢复后最终得到无噪声的点云。2)选择包含点云数量最多的连通域。如果存在极端的情况，如图 8 所示，由于物体点云紧密相邻，而噪声点之间的间隔稀疏，当点云映射到图像上时，噪声点对应的连通域面积大于物体点云对应的连通域面积。此时如果依然采取选择最大连通域的方法，会得到错误的结果。因此在计算完映射图像的连通域后，需增加一步操作，即计算每个连通域中点云的数量。如图 8(b)所示，噪声点区域对应的点云数量为 8 个，物体点云区域对应的点云数量为 26 个，因此选择包含点云数量较多的连通域，并滤除其他区域，如图 8(c)所示，最终经过点云恢复后得到无噪声的点云。

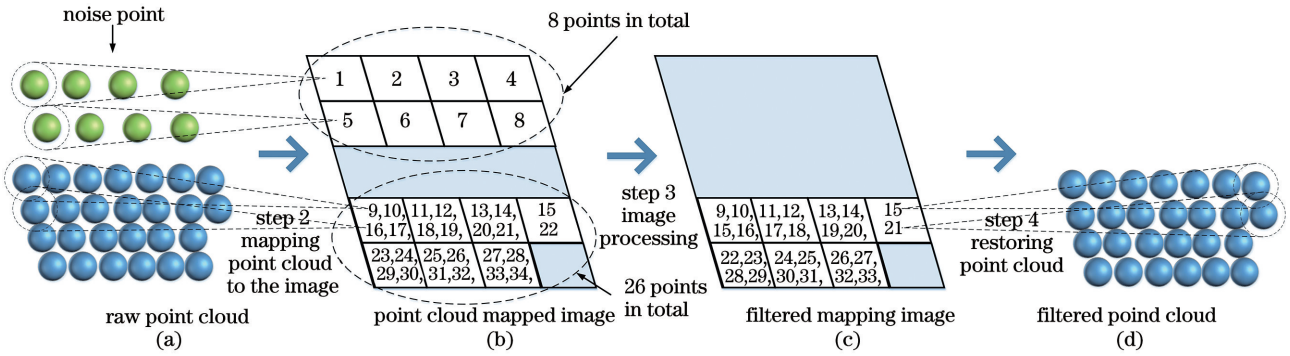

图 8 图像处理示意图(选择包含点云数量最多的连通域)

Fig. 8 Schematic of image processing (selecting the connected area with the largest number of point clouds)

如图 6 所示，经过 step 3 图像处理后，噪声点被滤除，最后只剩下物体点云对应的点云序号，最终根据点云序号重新恢复点云。

总而言之，在大多数情况下采用选择面积最大的连通域的方法更为高效快速。但在物体点云所占面积较小或者噪声点较多的情况下，可能会出现噪声点云对应区域面积大于物体点云对应区域面积的情况，在这种情况下通过进一步计算连通域中包含的点云数量，选择包含点云数量较多的连通域。虽然第二种方法较第一种方法多了一步，但计算速度依然较快，不会对算法的速率造成太大影响。

3.5 恢复点云

根据映射图像中对应的点云序号，在原始点云中找到和其序号一致的点即为物体点云。因此，恢复出映射图像中对应的点云序号，即得到了无噪声的点云。

如图 6 中 step 4 所示，经过图像处理后映射图像对应的点云序号有 1～5，序号为 6 的点被滤除后为空。在原始的点云(浮点型)中，找到序号为 1～5 的点，得到滤波后的点云(浮点型)，即为物体点云，最终将映射在图像中的点云从二维坐标系重新恢复为三维坐标系。

同理，如图 7、8 中 step 4 所示，将映射图像中对应的点云重新恢复为立体点云。

4 分析与讨论

4.1 实验结果分析

为了验证所提算法的有效性，选取一个玩偶作为三维重构的目标，如图 1 所示。图 2、3 为待测物体重构出来的点云正视图和侧视图，可以看到，物体点云附近存在大量的散乱噪声点和块状噪声点。

点云预处理(step 1)：为了避免对后续处理的影响，首先采用 k 最近邻滤除算法对点云进行散乱噪声点的滤除，滤除后的结果如图 9(a)所示，这时散乱噪声点已经被滤除干净，只剩下块状噪声点。从图 9(a)可以看到，块状噪声点和物体点云没有粘连，能够明确分割开来，然后对点云进行坐标转换和数据类型转换，使其能够映射到图像上。

将点云映射到图像上(step 2)：通过计算点云坐标沿 x 轴和 y 轴的最大值和最小值，建立一张以点云原点和以 x 轴和 y 轴为方向的映射图像($H\times W$)。将点云中的每个点都映射到图像上，并且标记图像每一点对应的点云数量和点云序号，最终结果如图 9(b)所示。

图像处理(step 3)：从图 9(b)可以明显看到，物体点云映射的图像区域具有断裂部位，若不采取任何处理，则在计算图像连通域时，会将物体点云映射的图像区域划分标记成不同的区域。因此采用形态学上的闭运算对映射图像进行处理，对断裂的部位

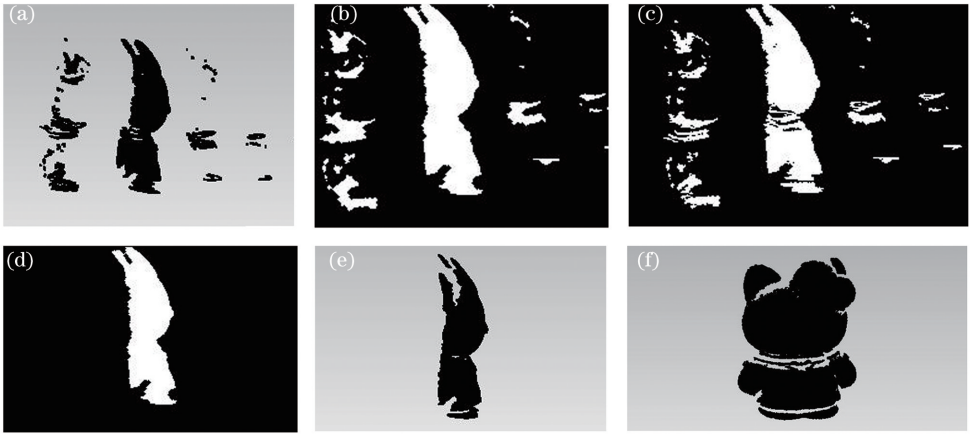

图 9 物体 1 的点云滤波过程。(a)k 最近邻滤波结果;(b)点云映射图像;(c)图像闭运算;(d)图像滤波处理;(e)点云滤波结果(侧视图);(f)点云滤波结果(正视图)

Fig. 9 Filtering process of point cloud for object 1. (a) Result of k nearest neighbor filtering; (b) point cloud mapped image; (c) image closing operation; (d) image filtering processing; (e) result of point cloud filtering (side view); (f) result of point cloud filtering (front view)

进行弥合,最终结果如图 9(c)所示。由于该玩偶重构的物体点云较大,映射到图像上的区域也相应较大,因此可以直接选择面积最大的连通域而不必选择包含点云数量最多的连通域。选定物体点云对应的区域后,滤除其他区域,最终结果如图 9(d)所示,只剩下物体点云对应的区域。

恢复点云(step 4):图 9(d)映射图像对应着点云的序号,根据这些序号,在图 2 点云中查找与之具有相同序号的点,保留具有相同序号的点而滤除其他点,最终就得到了不包含噪声点的物体点云,最终结果如图 9(e)和图 9(f)所示。

为了进一步验证点云滤波的效果,选择了另外一个物体进行三维重构,其过程和上述过程一样,其中物体2的滤波结果如图10(a)~(i)所示。对比

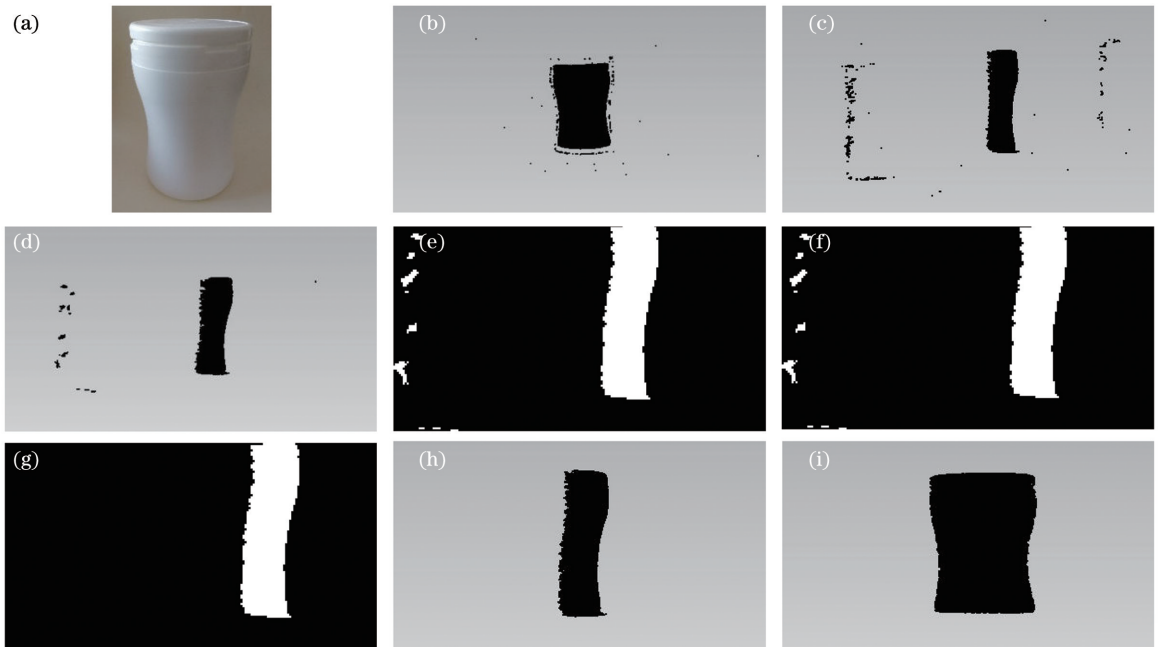

图 10 物体 2 的点云滤波过程。(a) 待测物体 2;(b) 点云正视图;(c) 点云侧视图;(d) k 最近邻滤波结果;(e) 点云映射图像;(f) 图像闭运算;(g) 图像滤波处理;(h)点云滤波结果(正视图);(i)点云滤波结果(侧视图)

Fig. 10 Filtering process of point cloud for object 2. (a) Object 2 to be measured; (b) front view of point cloud; (c) side view of point cloud; (d) result of k nearest neighbor filtering; (e) point cloud mapped image; (f) image closing operation; (g) image filtering processing; (h) result of point cloud filtering (front view); (i) result of point cloud filtering (side view)

图 10(c)和图 10(h)可以看到，所提算法在处理小的物体时依然具有较好的滤波效果。

选取一个具有噪声点较多的物体 3，如图 11(c)所示，该物体附近的噪声主要悬浮于该物体的前后，并且存在噪声点的数量多、体积大的特点。经过 k 最近邻滤波后，如图 11(d)所示，一些散乱的噪声点已经被去除，但一些大块的噪声点依然存在。将该点云映射到图像上，通过图像处理方法，大块的噪声点都已被去除，如图 11(h)、(i)所示。

通过上述三个例子，说明所提算法在点云中存在散乱的、大块的离群噪声点的情况下都有着较好的滤波效果。

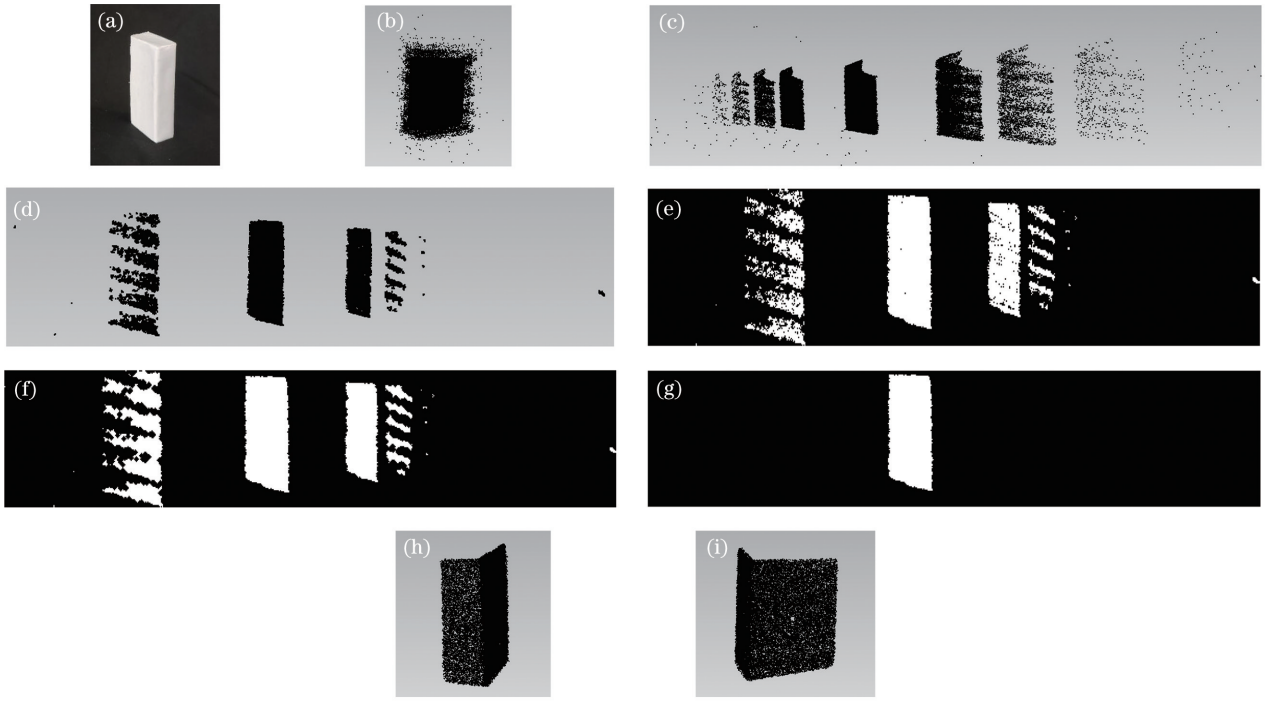

图 11 物体 3 的点云滤波过程。(a)待测物体 3；(b)点云正视图；(c)点云侧视图；(d)k 最近邻滤波结果；(e)点云映射图像；(f)图像闭运算；(g)图像滤波处理；(h)点云滤波结果(正视图)；(i)点云滤波结果(侧视图)

Fig. 11 Filtering process of point cloud for object 3. (a) Object 3 to be measured; (b) front view of point cloud; (c) side view of point cloud; (d) result of k nearest neighbor filtering; (e) point cloud mapped image; (f) image closing operation; (g) image filtering processing; (h) result of point cloud filtering (front view); (i) result of point cloud filtering (side view)

4.2 算法效率分析

为了进一步验证所提算法的效率，以这 3 个物体为例，分析总滤波时间与图像滤波各个步骤所耗费的时间。基于 MATLAB 2017a 软件编程实现所提算法，并在计算机(内存为 8 GB，主频为英特尔 i5-83002.3 GHz)上进行测试。

在图像处理过程中，采取了两种不同方法，分别为选择面积最大的连通域、选择包含点云数量最多的连通域，对比它们点云滤波的快慢。将点云滤波的时间分成两个部分，一个是 k 最近邻滤波所用的时间，一个是其他处理所用的时间。其他处理所用的时间是指除了 k 最近邻滤波时间外，其他过程所耗费的时间。表 1 和表 2 中计算所得的时间为平均 10 次实验的结果，并且 k 的大小设置为 20。在对比实验中，还对比了物体大小对点云滤波速率的影响。

由表 1 可知，在选择大物体情况下(即物体 1)，k 最近邻滤波所需时间为 0.5396 s，其他处理时间为 0.3689 s，说明所提算法将点云映射到图像上后，处理时间大大缩短，加快了算法的效率，并且总的耗费时间为 0.9085 s，达到了实时性的滤波效果。另外，在选择小物体情况下(即物体 2)，总的耗费时间为 0.1792 s，算法运行的速率更快。

对比表 1 和表 2，当在 step 3 中选择不同处理方式时，算法所耗费的时间也不一样。在同样选择大物体情况下，选择面积最大的连通域的方法所耗费的时间为 0.9085 s，选择包含点云数量最多的连通域的方法所耗费时间为 0.9516 s，说明在点云数量较多的情况下，第二种方法的效率相对较高。在同样选择小物体情况下，选择面积最大的连通域的方法所耗费时间为0.0611s，选择包含点云数量最

表 1 面积最大的连通域的方法所耗费时间

Table 1 Time consuming of largest connected area method

Model name	Number of raw point clouds	Number of filtered point clouds	Time consuming in k nearest neighbor filtering /s	Time consuming in remaining processing /s	Total time /s
Object 1	188925	167796	0.5396	0.3689	0.9085
Object 2	53838	53172	0.1181	0.0611	0.1792
Object 3	64933	31528	0.1616	0.2665	0.4281

表 2 包含点云数量最多的连通域的方法所耗费时间

Table 2 Time consuming of the connected area with the largest number of point clouds

Model name	Number of raw point clouds	Number of filtered point clouds	Time consuming in k nearest neighbor filtering /s	Time consuming in remaining processing /s	Total time /s
Object 1	188925	167796	0.5396	0.4120	0.9516
Object 2	53838	53172	0.1181	0.0615	0.1796
Object 3	64933	31528	0.1616	0.2743	0.4359

多的连通域的方法所耗费时间为 0.0615 s，说明在点云数量较少时，两种方法的处理速度几乎一致。

通过实验分析，所提算法能够在对噪声点进行有效滤除的情况下，能大大减少滤波时间，达到实时性效果。另外，在点云数量较少情况下，图像处理时，选择包含点云数量最多的连通域的方法更佳；而当点云数量较多情况下，图像处理时，选择保留面积最大的连通域的方法相对较快。

4.3 实验效果对比

为了验证所提算法的滤波效果，选取其他 3 种滤波算法进行对比，分别是 k 最近邻滤波算法、半径滤波算法、体素滤波算法。图 12(a)～(d)分别为所提算法、k 最近邻滤波算法、半径滤波算法、体素滤波算法对物体 1 的滤波效果。从图 12 可知：所提算法能够干净地滤除散乱和大块的悬浮噪声点；k 最近邻滤波算法和半径滤波算法能够滤除散乱的噪声点，但对一些大块的噪声点没有办法滤除；在存在大量噪声的情况下，体素滤波算法也无法去除一些散乱的噪声点，而且主体点云也被滤除一部分，对离群点滤除效果不佳。图 13(a)～(d)分别为所提算法、k 最近邻滤波算法、半径滤波算法、体素滤波算法对物体 2 的滤波效果。从图 13 可知：所提算法能干净地滤除散乱点和小的块状悬浮噪声；k 最近邻滤波算法、半径滤波算法和体素滤波算法都没办法去除块状噪声。图 14(a)～(d)分别为不同算法对物体 3 的滤波效果。从图 14 可知：物体 3 的悬浮噪声点很多而且体积很大，因此k最近邻滤波算法、

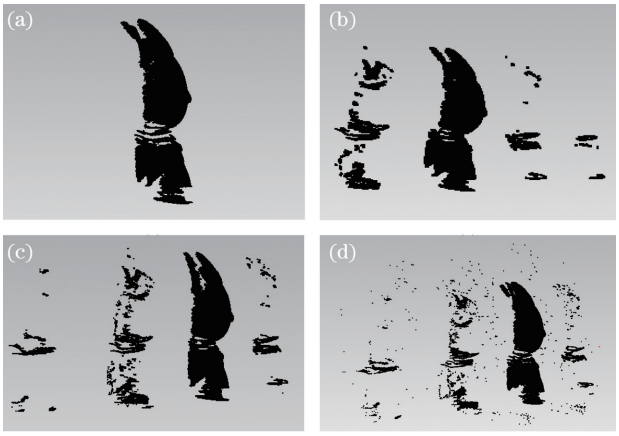

图 12 不同算法对物体 1 的滤波结果。(a)所提算法；(b)k 最近邻滤波算法；(c)半径滤波算法；(d)体素滤波算法

Fig. 12 Filtering results of different algorithms for object 1. (a) Proposed algorithm; (b) k nearest neighbor filtering algorithm; (c) radius filtering algorithm; (d) voxel filtering algorithm

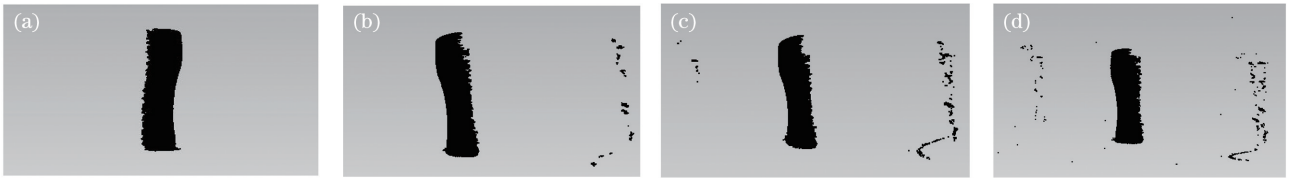

图 13 不同算法对物体 2 的滤波结果。(a)所提算法；(b)k 最近邻滤波算法；(c)半径滤波算法；(d)体素滤波算法

Fig. 13 Filtering results of different algorithms for object 2. (a) Proposed algorithm; (b) k nearest neighbor filtering algorithm; (c) radius filtering algorithm; (d) voxel filtering algorithm

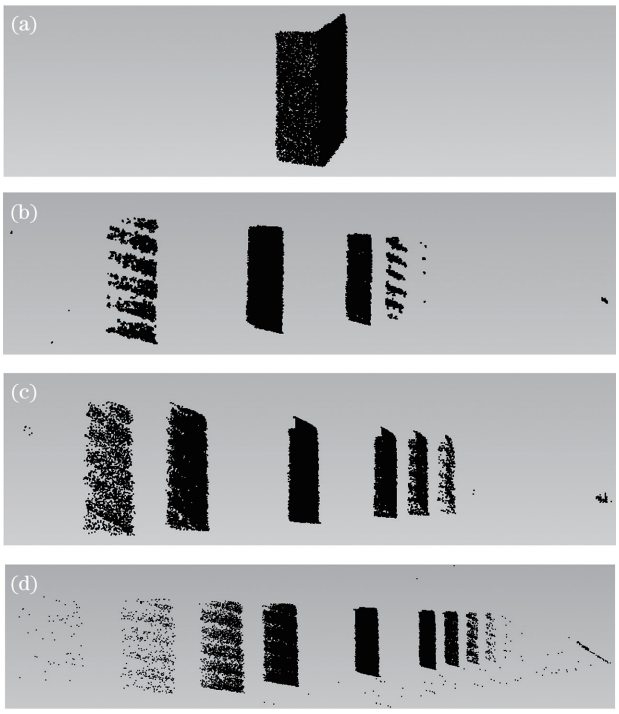

图 14 不同算法对物体 3 的滤波结果。(a)所提算法;(b)k 最近邻滤波算法;(c)半径滤波算法;(d)体素滤波算法

Fig. 14 Filtering results of different algorithms for object 3. (a) Proposed algorithm; (b) k nearest neighbor filtering algorithm; (c) radius filtering algorithm; (d) voxel filtering algorithm

半径滤波算法和体素滤波算法都无法滤除噪声点;而所提算法通过点云映射的方式将其投影到图像平面,计算图像中包含点云数量最多的连通域,达到滤除其他噪声部分只保留主体点云的目的,因此滤除效果较好,最终不再包含有悬浮的块状噪声点。

通过上述实验表明,在点云存在大量散乱点、体积大的离群噪声点的情况下,由于体积大的离群噪声点干扰太大,k 最近邻滤波算法、半径滤波算法和体素滤波算法都没有办法完全去除。而所提算法通过将点云映射到图像平面后,能够区分主体点云和离群点云的关系,通过图像处理方式快速定位噪声点云并去除,具有优越的点云滤波处理能力。

4.4 实验精度对比

计算点云的滤波精度,表达式为

$$P=\frac{N_{\mathrm{TP}}}{N_{\mathrm{TP}}+N_{\mathrm{FP}}},\tag{4}$$

式中:N_{TP} 为把无噪声的点判定为无噪声点的点云数量;N_{FP} 为把噪声点判定无噪声点的点云数量。

表 3 为物体 1 的滤波结果,可知:体素滤波算法的速度最快,但 N_{TP} 值较低、N_{FP} 值较高,因此滤波后点云的精度不高,只有 13.977%;k 最近邻滤波算法和半径滤波算法的效果较好,精度分别达 89.717%和 89.314%,而且 k 最近邻滤波算法比半径滤波算法的速度快,只需 0.486 s 时间;经过所提算法滤波后,N_{TP} 值为 167736,和 k 最近邻滤波算法和半径滤波算法的 N_{TP} 差不多,但 N_{FP} 相比要小很多,只有 60,因此所提算法的精度达 99.964%,实现了点云的高精度滤波,而且速度和其他两个算法相比差别不大,速度能达 0.909 s。表 4 为物体 2的滤波结果,可知:体素滤波算法的速度虽然最

表 3 物体 1 的滤波效果分析

Table 3 Filtering effect analysis of object 1

Parameter	Proposed algorithm	k nearest neighbor filtering algorithm	Radius filtering algorithm	Voxel filtering algorithm
Number of filtered point clouds	167795	187028	187940	90878
Time consuming in filtering /s	0.909	0.486	0.755	0.081
N_{TP}	167736	167796	167857	12702
N_{FP}	60	19232	20083	78176
P /%	99.964	89.717	89.314	13.977

表 4 物体 2 的滤波效果分析

Table 4 Filtering effect analysis of object 2

Parameter	Proposed algorithm	k nearest neighbor filtering algorithm	Radius filtering algorithm	Voxel filtering algorithm
Number of filtered point clouds	53172	53417	53637	24428
Time consuming in filtering /s	0.341	0.134	0.213	0.025
N_{TP}	53166	53172	53178	3518
N_{FP}	6	245	459	20910
P /%	99.988	99.541	99.144	14.402

快，但是精度最差；k 最近滤波算法和半径滤波算法的滤波精度分别为 99.541%和 99.144%，滤波精度较高并且差异不大，而且时间也分别达 0.134 s 和 0.213 s，速度较快；所提算法的滤波精度为 99.988%，但和 k 最近邻滤波算法和半径滤波算法的精度相比效果不明显，因为该物体的点云只有少量的散乱点和块状噪声点，速度也很快，时间达到了 0.341 s。图 11(c)为物体 3 的点云，包含有大量的散乱点和块状噪声点，分布在主体点云的前后位置。由表 5 可知：k 最近邻滤波算法、半径滤波算法和体素滤波算法的点云精度分别为 61.465%、49.486%、44.647%，滤波后的点云精度都不高；在存在大量离群点噪声的情况下，所提算法采用图像处理方式，能够快速准确地将噪声点去除，处理时间达 0.427 s，而且滤波后的点云精度达到 99.686%，远超过其他 3 种算法的点云精度。

表 5 物体 3 的滤波效果分析

Table 5 Filtering effect analysis of object 3

Parameter	Proposed algorithm	k nearest neighbor filtering algorithm	Radius filtering algorithm	Voxel filtering algorithm
Number of filtered point clouds	31528	51294	63915	61061
Time consuming in filtering /s	0.427	0.165	0.273	0.028
N_{TP}	31429	31528	31629	27262
N_{FP}	99	19766	32286	33799
P /%	99.686	61.465	49.486	44.647

因此，所提算法在存在大量离群点噪声的情况下，还能在保证滤波速度的情况下，大幅度地提升点云的滤波精度，十分适用于实际的工业应用。

5 结 论

提出了一种基于图像处理的点云滤波算法。通过对点云进行预处理，将点云映射到图像上，利用图像处理方法来处理噪声点，即将点云从三维空间降维到二维空间，降低了算法的复杂性，有效提升了点云滤波的速度。实验结果表明，所提算法对离群的噪声点滤除效果较好，并且能实时性地对点云噪声进行处理。

参 考 文 献

[1] Zhang S. High-speed 3D shape measurement with structured light methods: a review[J]. Optics and Lasers in Engineering, 2018, 106: 119-131.

[2] Song Z. Handbook of 3D machine vision: optical metrology and imaging[M]. Wales: CRC Press, 2016.

[3] Rusu R B, Cousins S. 3D is here: point cloud library (PCL)[C]//2011 IEEE International Conference on Robotics and Automation, May 9-13, 2011, Shanghai, China. New York: IEEE Press, 2011.

[4] Poddar S, Patra B K. Reduction inexecution cost of k-nearest neighbor based outlier detection method [M]//Ghosh D, Giri D, Mohapatra R N, et al. Mathematics and computing. Communications in computer and information science. 2018, 834: 53-60.

[5] Fleishman S, Drori I, Cohen-Or D. Bilateral mesh denoising[C]//ACM SIGGRAPH 2003 Papers, July 27-31, 2003, San Diego, California. New York: ACM Press, 2003: 950-953.

[6] Zheng Y Y, Fu H B, Au O K C, et al. Bilateral normal filtering for mesh denoising [J]. IEEE Transactions on Visualization and Computer Graphics, 2011, 17(10): 1521-1530.

[7] Liao Z P, Bai H P, Chen L. Improved denoising of point-sampled model based on bilateral filtering[J]. Computer Technology and Development, 2019, 29 (11): 42-46.
廖中平, 白慧鹏, 陈立. 基于双边滤波改进的点云平滑算法[J]. 计算机技术与发展, 2019, 29(11): 42-46.

[8] Zheng L J, Zhang J, Zheng L P. A fast implementation of bilateral filtering with voxel[J]. Video Engineering, 2018, 42(12): 11-17.
郑丽娟, 张杰, 郑丽萍. 基于体素的双边滤波快速实现[J]. 电视技术, 2018, 42(12): 11-17.

[9] Martin R R, Stroud I A, Marshall A D. Data reduction for reverse engineering[EB/OL]. [2020-08-11]. https://www.researchgate.net/publication/265775577_Data_reduction_for_reverse_engineering.

[10] Li R Z, Yang M, Ran Y, et al. Point cloud denoising and simplification algorithm based on method library [J]. Laser & Optoelectronics Progress, 2018, 55 (1): 011008.
李仁忠, 杨曼, 冉媛, 等. 基于方法库的点云去噪与精简算法[J]. 激光与光电子学进展, 2018, 55(1):

011008.

[11] Zhong Z P, Zhang J Z, Liang B. Point cloud denoising based on OPTICS clustering and improved bilateral filtering[J]. Modern Computer, 2020(10): 76-80.
钟志鹏, 张建州, 梁彪. 基于 OPTICS 聚类和改进双边滤波的点云去噪方法[J]. 现代计算机, 2020(10): 76-80.

[12] Almonacid J, Cintas C, Derieux C, et al. Point cloud denoising using deep learning [C] // 2018 Congreso Argentino de Ciencias de la Informática y Desarrollos de Investigación (CACIDI), November 28-30, 2018, Buenos Aires, Argentina. New York: IEEE Press, 2018.

[13] Zhao K, Xu Y C, Li Y L, et al. Large-scale scattered point-cloud denoising based on VG-DBSCAN algorithm [J]. Acta Optica Sinica, 2018, 38(10): 1028001.
赵凯, 徐友春, 李永乐, 等. 基于 VG-DBSCAN 算法的大场景散乱点云去噪[J]. 光学学报, 2018, 38(10): 1028001.

[14] Zhao J D, Yang F H, Liu A J. Near outlier detection of scattered point cloud [J]. Journal of Computer Applications, 2015, 35(4): 1089-1092, 1128.
赵京东, 杨凤华, 刘爱晶. 散乱点云近离群点识别算法[J]. 计算机应用, 2015, 35(4): 1089-1092, 1128.

[15] Nurunnabi A, West G, Belton D. Outlier detection and robust normal-curvature estimation in mobile laser scanning 3D point cloud data [J]. Pattern Recognition, 2015, 48(4): 1404-1419.

[16] Yang Y W, Li X G, Zhang Y P. Image based outlier detection algorithm for point cloud reconstruction[J]. Journal of Data Acquisition and Processing, 2018, 33(5): 928-935.
杨雨薇, 李幸刚, 张亚萍. 基于图像的重建点云离群点检测算法[J]. 数据采集与处理, 2018, 33(5): 928-935.

[17] Arvanitis G, Lalos A S, Moustakas K, et al. Real-time removing of outliers and noise in 3D point clouds applied in robotic applications [M] // Ronzhin A, Rigoll G, Meshcheryakov R. Interactive collaborative robotics. Lecture notes in computer science. 2017, 10459: 11-19.

[18] Ning X, Li F, Tian G, et al. An efficient outlier removal method for scattered point cloud data [J]. PLoS One, 2018, 13(8): e0201280.