

Article

Image Inpainting for 3D Reconstruction Based on the Known Region Boundaries

Hailong Yan ¹, Wenqi Wu ¹, Zhenghua Deng ², Junjian Huang ^{1,*}, Zhizhang Li ¹ and Luting Zhang ³

¹ College of Electronic and Information Engineering, Southwest University, Chongqing 400715, China; yhl00825@163.com (H.Y.); lifewhowwq@163.com (W.W.); jiaowozge@email.swu.edu.cn (Z.L.)

² Science and Technology Department, Chongqing University of Education, Chongqing 400065, China; dengzh@cque.edu.cn

³ People's Procuratorate of Beijing Municipality, Beijing 100078, China; zlt1980bj@163.com

* Correspondence: junjianhuang@swu.edu.cn

Abstract: Pointcloud is a collection of 3D object coordinate systems in 3D scene. Generally, point data in pointclouds represent the outer surface of an object. It is widely used in 3D reconstruction applications in various fields. When obtaining pointcloud data from RGB-D images, if part of the information in the RGB-D images is lost or damaged, the pointcloud data will be hollow or too sparse. Moreover, it is not conducive to the subsequent application of pointcloud data. Based on the boundary of the region to be repaired, we propose to repair the damaged image and synthesize the complete pointcloud data after a series of preprocessing steps related to the image. Experiments show that our method can effectively improve the restoration of the lost details of the pixel in the target area and that it will have the fuller pointcloud data after synthesizing the restored image.

Keywords: image inpainting; image processing; pointcloud

MSC: 54H30; 65D18



Citation: Yan, H.; Wu, W.; Deng, Z.; Huang, J.; Li, Z.; Zhang, L. Image Inpainting for 3D Reconstruction Based on the Known Region Boundaries. *Mathematics* **2022**, *10*, 2761. <https://doi.org/10.3390/math10152761>

Academic Editors: Rocio Gonzalez Diaz and Matthias Zeppelzauer

Received: 14 June 2022

Accepted: 30 July 2022

Published: 3 August 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Image inpainting refers to a technology in which the information in the local area of an image is lost. The purpose of image inpainting is to restore this information. In the field of traditional image inpainting, there have been many methods, such as the fast marching method (FMM) proposed by Telea [1], which has a better inpainting effect for small-scale images with small pixel differences and semantically poor images with missing areas. However, for large areas, large pixel differences and language-rich images with missing areas are very lacking in repairing effects, incoherent chromaticity in repaired areas, blurred content, and missing texture information. Criminisi [2] proposed the block-based image inpainting method (BBIM). This method considers texture information compared with the FMM. If the information of the object to be repaired is more complex or involves areas where multiple texture structures intersect, there will be repair dislocation and incomplete pixel filling, and it will lead to missing or empty results in the generated pointcloud data.

The FMM and BBIM all first obtain the mask image according to the binarization of the damaged image, then repair the original image based on the mask image. The missing pixel information in the image's damaged area is not uniform, or the pixel value of the damaged boundary area is not far from the normal value; the set binarization threshold range is too small. This will cause the effective information extracted from the mask image to be incomplete. As a result, the repair is synsemantic and the information of the pointcloud data is lost.

From these conditions, in order to obtain complete pointcloud data for subsequent 3D reconstruction and other related applications, we propose the following optimization scheme.

(1) For large image inpainting, the original image is divided into blocks, and only the image blocks that need to be restored are selected for subsequent operations. This reduces repair time and increases efficiency.

(2) When the pixel value of the damaged area is uniform, the extracted mask image is relatively complete. It also can fully cover the required repair area. However, the image damage is not deliberately damaged, and the lost pixel information is often uneven. The mask image extracted at this time needs to undergo two expansion processes to make it cover the damaged area as much as possible.

(3) The mask image after the expansion process was combined with the original image for inpainting. There are often some subtle holes that are not completely repaired. At this time, even a second repair is already saturated. Based on this, image sharpening is performed on the area to be repaired in order to avoid disturbing the intact area. It also can enhance the information of the area to be repaired, and then the mask image is extracted by binarization for subsequent inpainting.

In this paper, we consider the characteristics of the image to be repaired with vertical scratches (pixel values decay with the center of the scratches to both sides), which is not reflected in the original method. Secondly, in view of the shortcomings of the basic method, such as long repair time and low repair efficiency, we propose an image block parallel repair method to speed up the image repair speed and improve the repair efficiency.

2. Related Works

Bertalmio [3] and others first proposed the concept of image inpainting. The traditional image inpainting method mainly performs speculative inpainting based on the connection and continuity between pixels. In recent years, the research of image inpainting technology can be mainly divided into two methods: pixel-based and block-based.

The pixel-based method is suitable for repairing small-sized damaged images. This method mainly includes the partial differential equation (PDE) method, the variational method, and the interpolation method based on known information. The PDE method uses the edge features of the known area to estimate the iso-illuminance line direction and uses the push method to transmit the information along the iso-illuminance line direction to the area to be repaired. The BSCB method [3] and curvature diffusions repair method [4] were used. The variation-based method transforms the image inpainting process into a functional maximal value variational problem [5–8]. The interpolation method, based on known information, is to import the pixel value of the complete region of the image into the relevant equation. The calculated value is then used to update the point to be filled [1].

One of the methods based on block images is to obtain the boundary of the damaged area, sorting by the priority value of all points on the boundary. To employ this method, set up a module region with the highest priority value as the center. Match the template area with the original image area to obtain the block with the minimum mean square error. Finally, input the texture and structure information together to update the confidence value [2]. Based on this method, Luo [9] improved it and proposed a repair algorithm for depth image. Xu [10] proposed two sparse concepts, which distinguish between structural repair and texture repair. It uses the non-zero similarity sparsity of the repaired region and its adjacent region to obtain the reliability, and it improved the repair sequence. Anamandra [11] also introduced the gradient and its corresponding log value based on the selection of priority, achieving good repair effects. Huang [12] introduced the affine transformation matrix to realize geometric transformation of sample blocks, effectively improving the image repair effect.

Pointcloud data is widely used in the 3D reconstruction of objects due to its good shape expression ability. It also can reflect the 3D information of objects [13]. In the methods for generating pointclouds by using the RGB-D dataset [14,15], we see that if the image is damaged or the pixels are lost, the pointcloud data will cause holes or missing areas. In order to accurately restore the morphological characteristics of the scene, filling the pointcloud holes is also one of the basic tasks of the subsequent 3D reconstruction. The

existing research methods for pointcloud hole filling mainly include three methods: based on known information, based on matching, and based on learning.

The method based on known information uses the geometric features of existing pointcloud data to complete the whole pointcloud filling. For example, Tu [16] proposed a method based on Kriging interpolation, using the known points around it as a reference to fill the missing pointcloud. However, if the missing area of the pointcloud is too large, the geometric characteristics of the missing area cannot be estimated. Thus the missing area cannot be supplemented.

The core idea of the matching-based method is to use the model in the known pointcloud database to match the missing area of the pointcloud to achieve the completion task. By using both the deformation method [17] and the seamless stitching of triangular meshes based on Poisson equation [18], we complete the hole filling by triangularizing the cluttered pointcloud data to synthesize the geometric shape to be more consistent with the input. Li [19] proposed a method for repairing a pointcloud based on triangular meshing, which is to establish a tetrahedron by combining 3D curves and damaged pointclouds and then optimizing the tetrahedron for 3D curves. Finally, the mesh surface is extracted from the tetrahedron for pointcloud filling.

The learning-based approach is the recent widespread use of deep learning to process pointclouds directly [20,21]. Using neural networks to learn the characteristics of pointcloud data, and then directly filling the missing pointcloud has become a popular field of pointcloud repair research. For example, Qi proposed to use the PointNet network structure [22] to directly apply the deep learning network structure layer to the pointcloud data. The process of using deep learning for pointcloud repair is roughly to output the complete pointcloud directly after processing the pointcloud data to be repaired. On the basis of the literature [22], a PointNet++ network structure [23] is proposed, which is used for the processing of pointcloud data restoration. In the method of using RGB-D images to synthesize pointcloud, if the RGB image or depth image is damaged, it will also cause pointcloud holes. For the missing area of depth image, Zhang [24] only needs to input an RGB image and a depth image, which can fill the missing information of any depth map and achieve good results. It's also beneficial to the 3D reconstruction of pointcloud data based on the RGB-D dataset. Another method is the deep Laplacian pyramid network based on deep image enhancement using CNN, which has achieved good results by reducing the noise and holes of deep image in a cascading way [25]. This paper is based on the method of the pointcloud formed by the RGB-D dataset [14,15], through the repair of image pixel-loss area, the hole filling of the final pointcloud. Through the final repair effect, the practicability of the method is verified.

3. Base Algorithm Overview

3.1. Fast Marching Method

Let Ω be the area to be repaired, and $\delta\Omega$ be the boundary of the area. Take a point p on regional boundary, and we specify p as the center to select the area that does not need to be repaired around p . The pixel value of point p is approximated by the pixel value inside the small neighborhood $B(\epsilon)$ with ϵ as the scale. Let q be a point in $B(\epsilon)$. Giving the pixel value $I(q)$ of q and the gradient value of $\nabla I(q)$, then we can use q to calculate the gray value of p point, such as Equation (1) [1]:

$$I_q(p) = I(q) + \nabla I(q)(p - q). \quad (1)$$

The calculation equation of pixel value of point p is:

$$I(p) = \frac{\sum_{q \in B_\epsilon(p)} \omega(p, q) [I(q) + \nabla I(q)(p - q)]}{\sum_{q \in B_\epsilon(p)} \omega(p, q)}. \quad (2)$$

The weight function $\omega(p, q)$ is used to limit the contribution of each pixel in the neighborhood. The detailed information of the image edge is transmitted toward the region

to be repaired through the $\omega(p, q)$. In this method, take the point p as the center and the size of Ω is taken as the neighborhood of the area to be repaired. Then the gradient of p is calculated through this neighborhood. The gradient of p is the average of the largest gradients in the previous points. Finally, the information of the image is transmitted to the area that needs to be repaired according to the iso-illumination line $\bar{c}^\perp(p)$:

$$\omega(p, q) = \sqrt{\frac{\pi}{2}} \frac{\mu}{|p - q|} \exp\left(-\frac{\mu^2}{2\varepsilon^2} |\bar{c}^\perp(p) \cdot (p - q)|^2\right). \tag{3}$$

The repair process needs to use Equation (1) to repeatedly traverse all discrete points, increasing the distance of the initial boundary $\partial\Omega$. The FMM method process is shown in Figure 1.

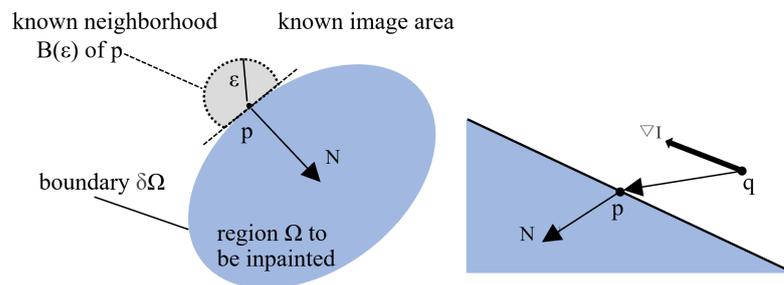


Figure 1. FMM process.

3.2. The Block-Based Image Inpainting Method

The BBIM proposed by Criminisi [2] uses the known area of the image to fill the area to be repaired. It also determines the filling order according to the calculation priority. The filling content is a pixel block of a custom size.

Divide the image to be repaired into a known area Φ and a region Ω to be inpainted in Figure 2a. Set $\delta\Omega$ as the front of fill area. In Figure 2b, taking point p ($p \in \delta\Omega$) as the center, the region defined by ψp . With ψp as the target matching region, match similar regions on the boundary between two textures in a known region, such as $\psi q'$, $\psi q''$ in Figure 2c. Finally, select the best matching region for filling in Figure 2d.

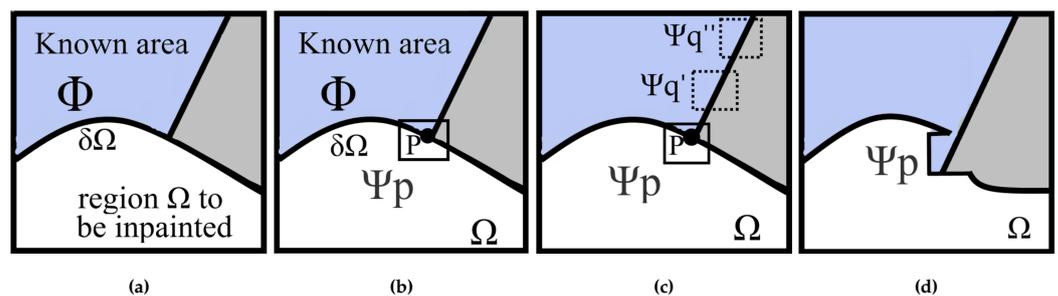


Figure 2. Structure propagation by known area texture synthesis. (a) Input. (b) Determine the point to be repaired. (c) Find the best matching block. (d) Fill the neighborhood to be repaired.

The method for calculating edge contour filling priority is given by

$$P(p) = C(p) \cdot D(p). \tag{4}$$

From Figure 3, the direction pointed to by n_p represents the normal tangential direction of the edge contour, ∇I_p^\perp is a 90° rotation of the pixel gradient direction of point p . Through these two variables, we can get the confidence term and data term:

$$C(p) = \frac{\sum_{q \in \Psi_p \cap \bar{\Omega}} C(q)}{|\Psi_p|}, D(p) = \frac{|\nabla I_p^\perp \cdot n_p|}{a}. \tag{5}$$

In Equation (5), $|\psi p|$ is the area of ψp , a is normalizing factor and its ordinary value is 255, $\nabla I_p^\perp = (-I_x, I_y)$, I as the image of binaryzation, and I_x and I_y are the gradients in the x and y directions. The final available $n_p = (I_x, I_y)$. The maximum value of $P(p)$ obtained from Equation (4) is selected as the center, and the scale is $\Psi_{\hat{p}}(n \times n)$.

Select the region $\Psi_{\hat{q}}$ which has the minimum mean squared error with $\Psi_{\hat{p}}$ in the known regions of the image.

Finally, texture and structure information is passed and $C(p)$ is updated. This process occurs when $\Psi_{\hat{p}}$ is replaced by a new pixel value, and $c(p)$ is constantly updated, as shown in Equation (6):

$$C(p) = C(\hat{q}). \tag{6}$$

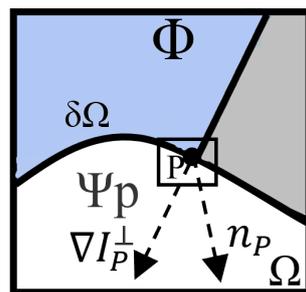


Figure 3. The process diagram of calculating the filling priority of image edge contour.

4. Related Methods Analysis

4.1. Image Preprocessing Methods

Image Dilation

Image binarization occurs when the pixel values of the damaged area and other areas are 255 or 0. The purpose of image dilation is to enlarge and thicken the area to be repaired. In this way, the repair failure caused by an uneven boundary pixel value can be alleviated.

Dilation Process

$$A \oplus B = \{x | (B)_x \cap A \neq \emptyset\} \tag{7}$$

Define the Expansion operator as \oplus .

B is used to expand image A , where B is A convolution template or convolution kernel. The convolution calculation is carried out between template B and image A , and every pixel in the image is scanned. The "AND" operation is performed with the template element and binary image element. If both are 0, the target pixel is 0; otherwise, it is 1. Thus, the maximum value of pixels in the covered area of B is calculated. And the pixel value of the reference point is replaced by this value. Therefore, the expansion result is the expansion and thickening of the white highlighted area. In order to cover the boundary of the region to be repaired in the binary image, we select the full 1 matrix with the dilation convolution kernel of 5×5 .

Dilation Convolution Kernel

$$\begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{bmatrix} \tag{8}$$

Image Sharpening

In the differential operation of image sharpening, the Laplace operator is commonly used [26]. As a linear quadratic differential operator, the Laplace operator has anisotropy and displacement invariance, which satisfies the sharpening requirements of image boundaries with different orientations.

For the digital image, the second-order partial derivative can be expressed by the second-order difference approximation, so the Laplace operator is defined as

$$\nabla^2 f(x, y) = f(x + 1, y) + f(x - 1, y) + f(x, y + 1) + f(x, y - 1) - 4f(x, y), \quad (9)$$

where $f(x, y)$ represents the pixel value of a image.

According to Equation (9), the Laplacian operator is based on the change of one pixel relative to four adjacent pixels,

$$\begin{aligned} g(x, y) &= f(x, y) - \nabla^2 f(x, y) \\ &= 5f(x, y) - f(x + 1, y) + f(x - 1, y) - f(x, y + 1) - f(x, y - 1) \end{aligned} \quad (10)$$

where $g(x, y)$ represents the pixel value sharpened by the Laplacian operator.

The mask image cannot completely cover the damaged part, and the binarization operation has a limited range of distinguishing pixel values. With the help of image sharpening, the purpose is to make the blurred image clear. It also can enhance the edges and contours of the image details, and it enhance the grayscale contrast so the image that has been repaired once can be extracted through the binarization operation, again for image inpainting. Because the pixel value of the image after sharpening has been changed, we only use the mask image extracted after sharpening to combine with the original image to be repaired before repairing.

Image Sharpening Convolution Kernel

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad (11)$$

Image Block

For the processing of large images, due to the limitation of the repair method, the efficiency has reached the limit after the first repair. However, there is still a small part of the area that has not been completely repaired. We use multi-thread method to process the original image by region. Only the small image regions to be repaired were extracted and repaired again. Eventually, the regions are regrouped. This avoids recovery limitations due to repair efficiency and speeds up repair.

4.2. Implementation Details

First, the image to be repaired is binarized to extract the mask image. The mask image segmented the area to be repaired. The image dilation algorithm is used to expand the area to be repaired in the mask image twice. The original image is repaired by FMM or BBIM. Sharpen the image obtained in one repair and then divide it into blocks. Then fill in the small area to be repaired and restore it in blocks to obtain the final image restoration result. In the end, convert the final restoration result into pointcloud data to view the restoration result of pointcloud data.

4.3. Image Dilation Results

After using the FMM or BBIM method to repair the image, there are still small areas not completely repaired on the boundary, as shown in Figure 4e. But the repaired results have reached the limit of FMM and BBIM methods. Effective inpainting areas cannot be extracted from mask again. According to the size of the expansion convolution kernel we set up and the experimental results. The mask, after expanding twice, can better express the region to be repaired. The method processing flow proposed in this paper is shown in Figure 5.

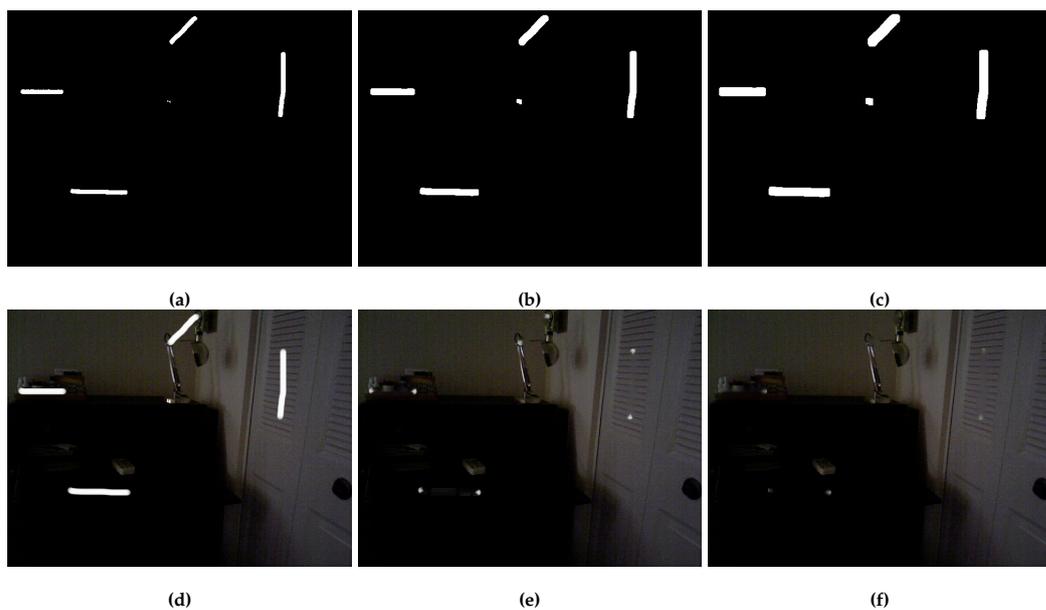


Figure 4. Results of the FMM combined with the image dilation. (a) Original mask. (b) The first dilation mask. (c) The second dilation mask. (d) Original image. (e) The first dilation result. (f) The second dilation result.

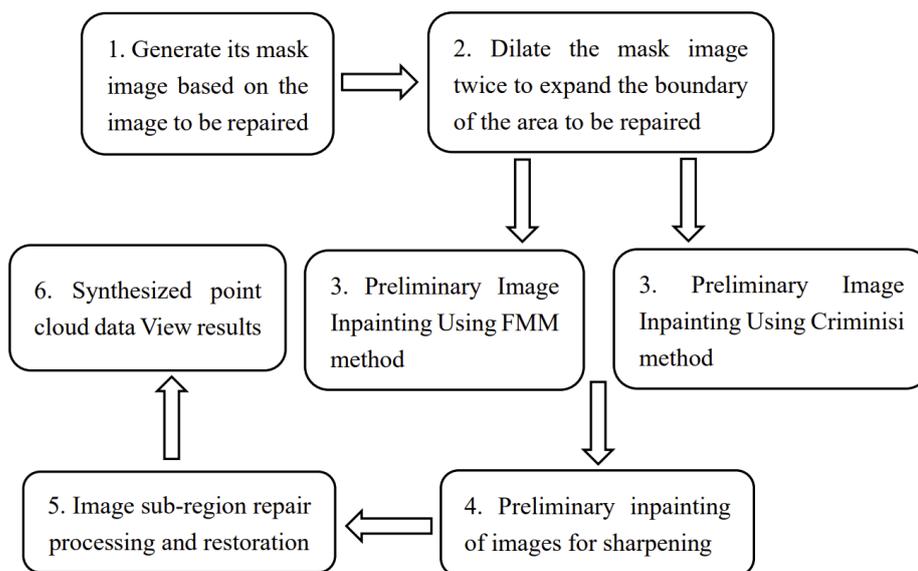


Figure 5. The flow chart of the method proposed in this paper.

It can improve the efficiency of repairing. From the result in Table 1, it can be concluded that the PSNR of the repair results after our processing is improved by 1.92 dB.

Table 1. PSNR results of image dilation.

Figure 4:	(e)	(f)
PSNR:	30.87	32.79

4.4. Image Sharpen Results

Figure 6a is the result of the our method combined with the BBIM proposed by Criminisi.



Figure 6. Image sharpening results. (a) Input. (b) Result.

It can be seen that most of the damaged parts of the image have been repaired and the texture information of the repaired area has been restored more efficiently. However, the characteristic of the edge of the image is that the grayscale change on the edge is relatively gentle, whereas the grayscale change is faster on both sides of the edge. Consequently the gradient value is large. It is usually the part with local discontinuity and the most significant change in brightness. It leads to the fact that there are still subtle parts of the damaged boundary that have not been completely repaired. At this time, the edge information of the area to be repaired is enhanced after the image-sharpening process. Its detailed features are highlighted, so that the mask image can be extracted after further binarization in Figure 6b.

4.5. Image Block Results

Although the unrepaired edge information of the abovementioned sharpened image has been enhanced, the unrepaired area has reached the repair limit of the above two methods. If the mask image is extracted by binary operation again, no area to be repaired has been extracted, but there are still subtle areas not repaired completely.

In order to solve the above problems and speed up the repair, we propose a method to fill small areas to be repaired by using multi-threaded images. We divide the original image into small areas for processing, as shown in Figure 7. Next, the block image containing the damaged region is sharpened to enhance the boundary information. Then, it is repaired again, and the repaired result is shown in Figure 8c.

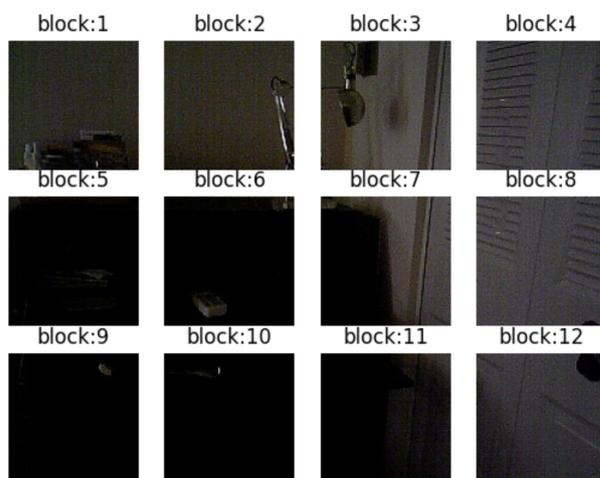


Figure 7. Image block result.

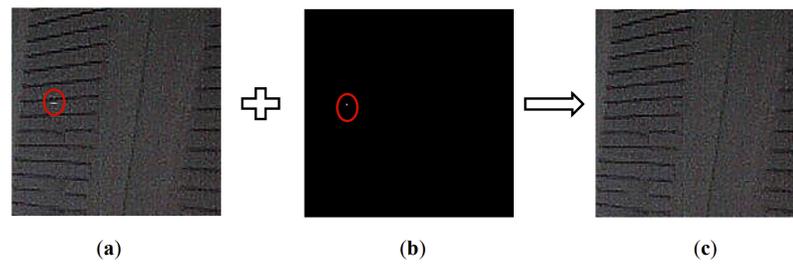


Figure 8. Block:4 process. (a) Block:4 input. (b) Block:4 mask image. (c) Block:4 result.

We divide the image into blocks such as Equation (12),

$$\begin{aligned} a &= \frac{m}{GCD} \\ b &= \frac{n}{GCD} \end{aligned} \tag{12}$$

where m and n represent the image width and height, GCD represents the greatest common divisor, b represents that the image is partitioned into rows b , and a represents that the image is partitioned into column a .

In the experiment, the size of the images we used was 640×480 . According to Equation (12), $m = 640$, $n = 480$, so the $GCD = 160$, and $a = 640/160 = 4$, $b = 480/160 = 3$, the input image is divided into 3×4 .

4.6. Pointcloud Visualization by RGB-D Odometry Method

RGB-D odometry method is one of the methods used to generate pointcloud data, and it is used to find the camera movement between two RGB-D images. Its input is a pair of RGB-D images, and its output is the motion in the form of rigid body transformation. The specific method is to convert the RGB image of 3 channels into the gray image of 1 channel, and then combine the depth image of a 1 channel with the camera to convert it into pointcloud [14,15].

First, an RGB-D Image is created by using a pair of color and depth images, and a proportional depth value d_{scale} is set. The depth value is scaled and then truncated. The color image is converted to a gray image and stored in the range of [0,1]. The depth image is stored in another floating point number and represents the depth value in meters.

Given the depth value d at pixel (i, j) , the corresponding 3D point is (X, Y, Z) ,

$$(X, Y, Z) = \begin{cases} X = \frac{(i - C_x) * Z}{f_x} \\ Y = \frac{(j - C_y) * Z}{f_y} \\ Z = \frac{d}{d_{scale}} \end{cases} \tag{13}$$

where focal length $(f_x, f_y) = (525.0, 525.0)$, optical center $(C_x, C_y) = (319.5, 239.5)$, and d_{scale} is 1000. Pixels (i, j) need to be converted from pixel coordinates to camera coordinates to display each point (X, Y, Z) in the pointcloud. According to the principle of similar triangles, $X/(i - C_x) = Z/f_x$, $Y/(j - C_y) = Z/f_y$, and Z is represented by the depth d of the corresponding point and the scaling coefficient d_{scale} .

Matrix Parameters of Camera

$$\begin{bmatrix} 525 & 0 & 319.5 \\ 0 & 525 & 239.5 \\ 0 & 0 & 1 \end{bmatrix} \tag{14}$$

5. Final Results

5.1. Image Inpainting

The common damage of images, such as vertical scratches, scratches of any shape, and dust, seriously affect the quality of images. The vertical scratch is a kind of continuous damage, strip shape. The pixel value of the damaged area is attenuated chord distribution, that is the pixel value of the damaged area decreases with the increase of the distance from the center area [27], as shown in Figure 9. Here, we artificially created images with vertical scratches, as shown in Figure 10a. Figure 9b is the magnification effect diagram of the damaged area boundary in Figure 9a.

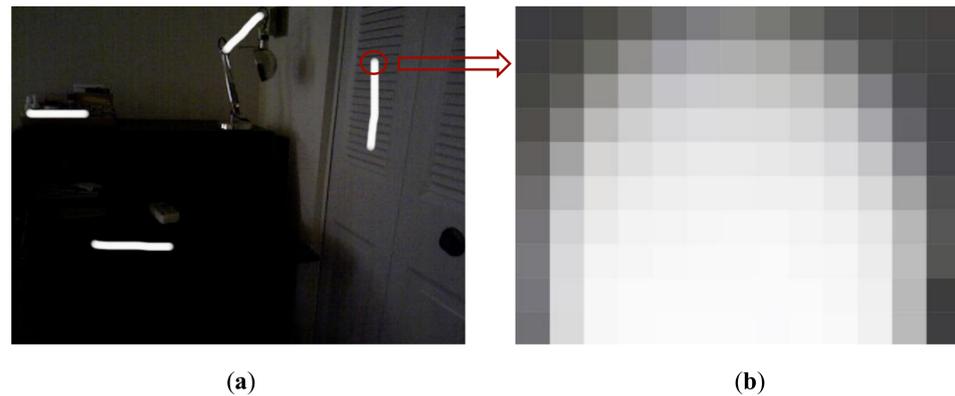


Figure 9. Input image. (a) Input. (b) Edge amplification result of damaged area.

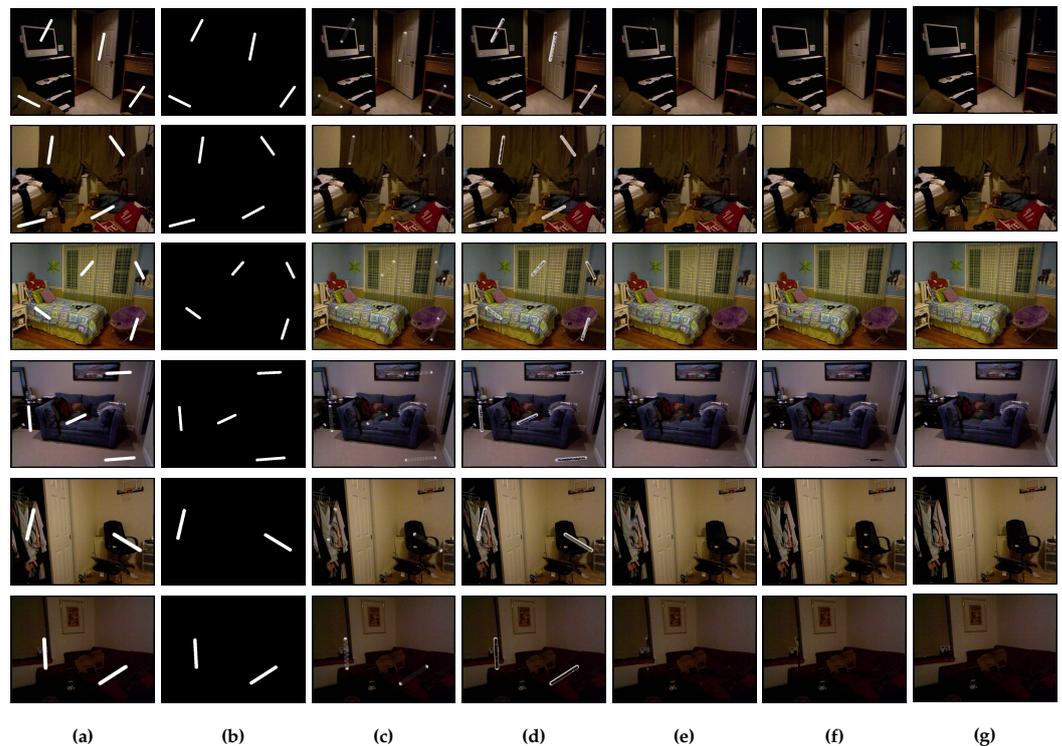


Figure 10. RGB image inpainting results. (a) Input. (b) Mask. (c) FMM results. (d) BBIM results. (e) Ours (based on FMM). (f) Ours (based on BBIM). (g) Ground truth.

This experiment is implemented in a laptop environment with an Intel(R) Core (TM) i5-9300H CPU@2.40GHz, 8.0GB memory and 64-bit Windows 10, using Python 3.6 as the algorithm platform.

In order to verify the effectiveness of our proposed method, the images used in this paper are mainly NYU-Depth V2 dataset and Redwood dataset images. The NYU-Depth V2

dataset contains 1449 annotated RGB images and depth images with the size of 640×480 . The image structure of Redwood dataset is the same as the NYU-Depth V2. We selected representative indoor scenes from these two datasets, such as living rooms, bedrooms, etc. The peak signal-to-noise ratio (PSNR) is used as the objective evaluation index of the restoration effect.

The experimental content is to use the improved method based on FMM and BBIM to repair the image, and compare with the original method to repair the image, and use PSNR to reflect the difference between the repair effect before and after the improvement.

The calculation method of PSNR is as follows:

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \|S(i, j) - J(i, j)\|^2 \tag{15}$$

$$PSNR = 10 \cdot \log_{10} \left(\frac{Z_{max}^2}{MSE} \right), \tag{16}$$

where m and n represent the image width and height, respectively, Z_{max} represents the maximum gray value of all pixels in the image, S represents the original image, and J represents the restored image.

Both the FMM method and BBIM method rely on known information of damaged region boundaries for image inpainting. But for the image to be repaired with vertical scratches, the pixel values of the boundary of the damaged area are attenuated chord distribution. When binaryzation is used to extract the mask image, the boundary is not completely extracted. In this way, the edge damaged areas that we did not extract during image inpainting are also treated as known area information. These pixel values, as known information of the damaged boundary, become a reference instead of the damaged region pixel values. So the final FMM method and BBIM method repair effect is not ideal, as shown in Figure 10c,d.

In view of the situation that the pixels of the image are lost due to the vertical scratches (as in the mask, the FMM method, and BBIM method), we use morphological analysis optimizations and update the masks by dilation and sharpening. After processing, the effect of image restoration is more ideal, such as in Figure 10e,f. We also use PSNR to represent the results of image inpainting, as shown in Table 2.

Table 2. RGB Image PSNR Results.

Serial Number	Input	Basic Method (FMM)	Basic Method (BBIM)	Ours (Based on FMM)	Ours (Based on BBIM)
1	18.602(+0)	+10.615	+2.971	+21.401	+20.132
2	17.791(+0)	+11.250	+3.646	+21.639	+21.644
3	20.884(+0)	+10.764	+4.683	+17.240	+13.468
4	18.760(+0)	+11.553	+5.168	+20.785	+15.233
5	19.654(+0)	+11.228	+4.841	+13.714	+12.492
6	18.666(+0)	+11.218	+6.448	+22.995	+22.014
Figure 10	(a)	(c)	(d)	(e)	(f)

We number the images in Figure 10 from top to bottom for 1, 2, 3, 4, 5, 6.

Because we use the pointcloud generated by RGB-D Odometry Method, the RGB image and depth image become two important factors affecting the pointcloud. In Table 2, PSNR is used as the evaluation standard of the RGB image inpainting results. In order to comprehensively consider the two factors, we add the same scratches as RGB images to the depth image of the input image in Figure 10a and repair them. The results of PSNR are used to evaluate the results of depth image repair, such as in Table 3. The results of Table 3 show that for the inpainting of depth images, the FMM method and BBIM method after the proposed method are more effective than the original basic method.

Table 3. Depth image PSNR results.

Serial Number	Input	Basic Method (FMM)	Basic Method (BBIM)	Ours (Based on FMM)	Ours (Based on BBIM)
1	19.262(+0)	+12.113	+4.807	+29.486	+27.857
2	18.778(+0)	+12.086	+4.950	+28.023	+30.264
3	20.824(+0)	+11.863	+5.305	+29.692	+28.392
4	19.276(+0)	+11.766	+5.376	+28.550	+29.577
5	21.352(+0)	+12.753	+5.230	+27.334	+28.087
6	20.616(+0)	+12.964	+5.418	+29.585	+30.061

Figure 10 (a) (c) (d) (e) (f)
 We number the depth image corresponding to the RGB image in Figure 10 from top to bottom for 1, 2, 3, 4, 5, 6.

5.2. Multithread Image Inpainting Result

The BBIM method is based on the principle of image block matching repair, and its basic idea is similar to that of FMM method. It introduces the concept of priority as a condition for filling pixel sequence selection. When repairing an image by the BBIM method, the first image block is selected from the area to be repaired and replaced by matching until the last image block of the area to be repaired is calculated. In order to shorten the repair time, we divide the image into blocks for multithreading repair.

According to the size of the input image, the number of input image blocks is calculated according to the principle of image block. The binary mask image processed by dilation method is also divided into the same blocks as the input image. In the basic BBIM method and FMM method, especially for block matching based on the BBIM method, it takes a lot of time to calculate the priority of filled blocks. In the process of image inpainting by using the BBIM method, a block with the highest priority is found according to the global image to fill the repair area. If the input image size is too large, it will also increase the calculation time for calculating the priority selection of filling blocks. In the local continuous region of the image, most of the pixels closer to the position have similar pixel information. Therefore, we divide the image to be repaired into several image blocks with the same size as the mask image, and repair these image blocks at the same time. Because the size of the image block becomes smaller and multiple repair areas are processed at the same time, the efficiency of image is effectively improved. We calculate the repair time of RGB images and depth images in Figure 10a, and the results are shown in Tables 4 and 5.

Table 4. RGB Image inpainting time results.

Serial Number	BBIM Results	Ours (Based on BBIM)
1	14.56	0.78
2	13.87	0.84
3	10.92	0.87
4	14.51	0.81
5	9.48	0.53
6	9.35	0.52

Unit: hours (d) (f)

The method of multi-thread repair in sub-regions is used, which improves the timeliness of the method and speeds up the repair speed.

Table 5. Depth Image inpainting time results.

Serial Number	BBIM Results	Ours (Based on BBIM)
1	12.16	0.68
2	17.60	0.51
3	14.37	0.69
4	16.03	0.65
5	7.52	0.51
6	11.48	0.53
Unit: hours	(d)	(f)

5.3. Generate Pointcloud Data Result

Our pointcloud generation method is based on the RGB-D odometry method. Pointclouds generated by using broken RGB images and depth images are shown in Figure 11a. The damaged area of the RGB image causes holes in pointcloud data. The pointcloud data repaired by using the FMM method and BBIM method are shown in Figure 11b,c. Because the boundaries of unextracted damaged areas are introduced in the repair process, and the boundaries of these damaged areas are used as reference information for repair, the repair effect is unsatisfactory. The broken RGB image we used in the process of generating pointcloud results in a hole in pointcloud data.

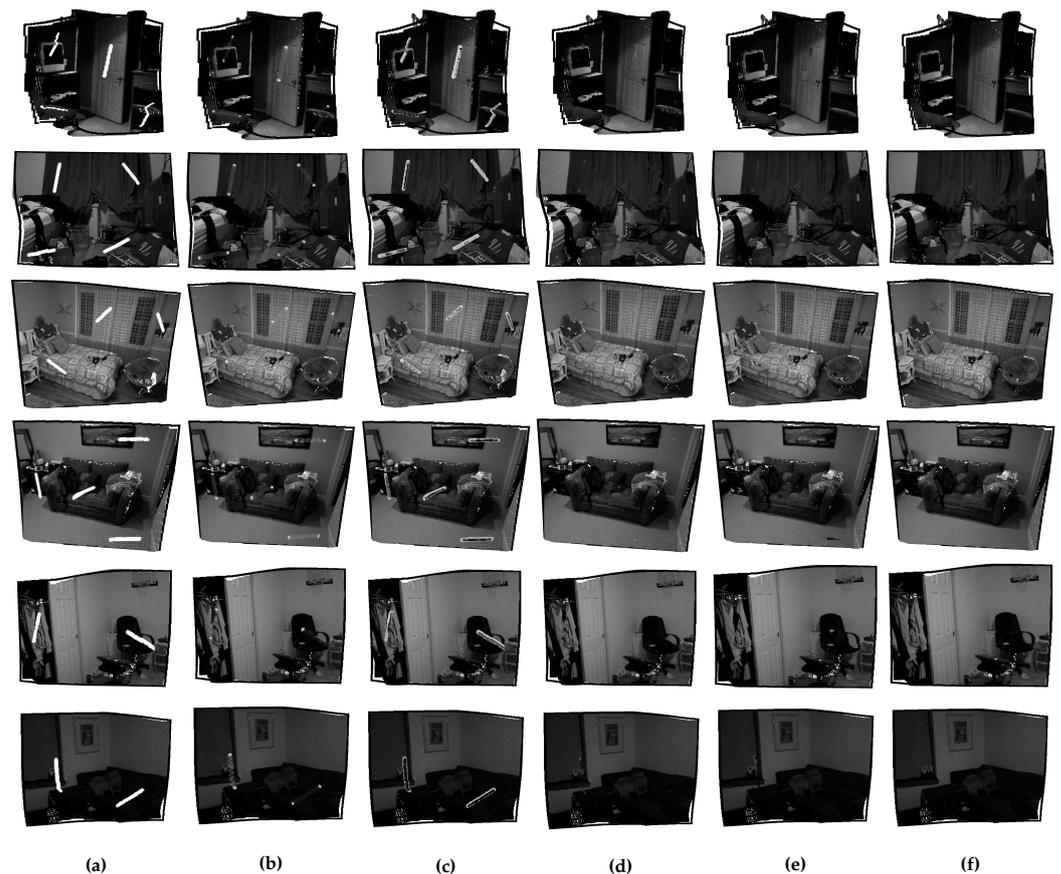


Figure 11. Effect of our simple post-processing. (a) The original pointcloud. (b) The pointcloud of FMM. (c) The pointcloud of BBIM. (d) Ours (based on FMM). (e) Ours (based on BBIM). (f) Ground truth.

Because the RGB image has scratches, the pointcloud generated by RGB-D will produce a pointcloud missing as shown in Figure 11a. Each pixel value in the depth image represents the plane distance between the scene object and the camera. If the depth image has scratches,

the pointcloud depth error shown in Figure 12a will occur. Figure 12b,c is the pointcloud synthesized by using the basic FMM and BBIM methods to repair the depth image. As the result of RGB image inpainting, the boundary of the area to be repaired in the depth image is not completely repaired, so the depth information of the boundary of the area will be wrong, which will affect the generated pointcloud.

Figures 11d,e and 12d,e show the pointcloud data processed by the proposed method combined with FMM method and BBIM method, and the repair effect is improved. However, it can be seen from the side view that even if the repair effect is improved compared with Figure 12b,c, there are still subtle incomplete parts in the scratch boundary area, which leads to some errors in the depth information of the boundary.

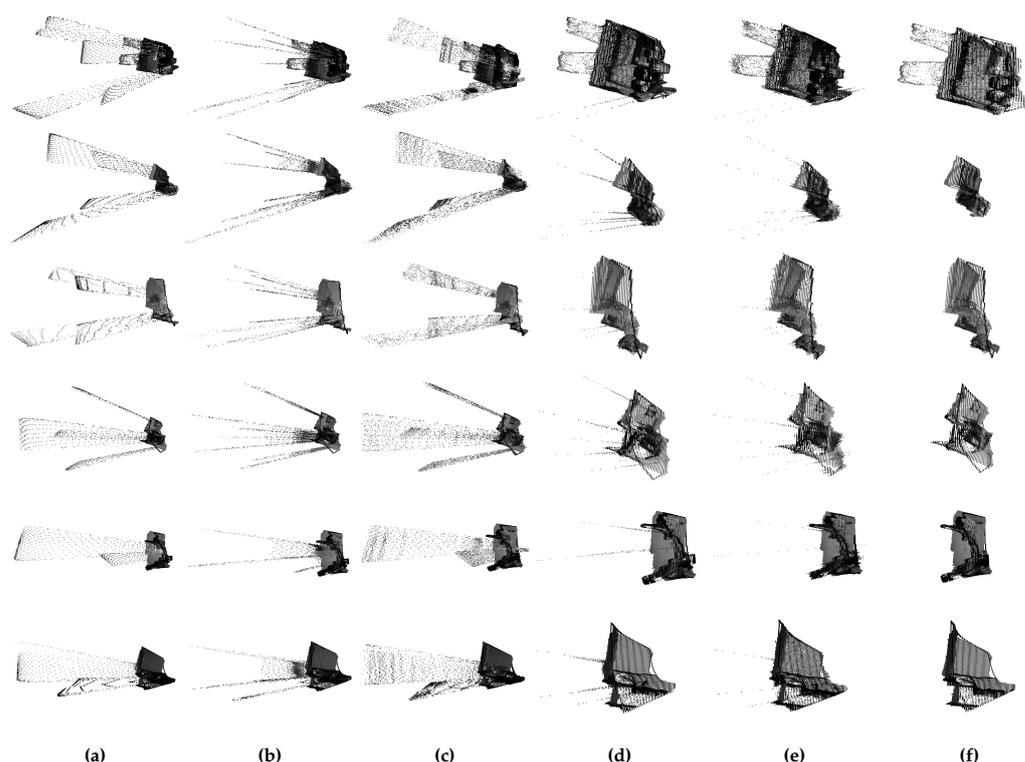


Figure 12. Left view of pointcloud. (a) The original pointcloud. (b) The pointcloud of FMM. (c) The pointcloud of BBIM. (d) Ours (based on FMM). (e) Ours (based on BBIM). (f) Ground truth.

6. Conclusions

We propose a new method for processing image inpainting, which is based on the FMM proposed by Telea and BBIM proposed by Criminisi, and which combines some image-processing methods, such as the dilation method for mask image, operations on sharpening, and image block processing of image.

We have proved that the above two repair methods combined with our method can achieve better repair results. It can also better reflect the texture information of the damaged area, and the repair speed is faster. The repaired pointcloud data fills in the information of the defective area, so that it can better reflect the three-dimensional space state of the object. It is more conducive to the extraction and segmentation of the target object when using the pointcloud data for 3D reconstruction and other applications.

7. Future Works

The purpose of pointcloud hole filling is to fill the sparse, incomplete or noisy and uneven data collected by 3D sensors. Applying high-quality pointclouds to 3D reconstruction and other fields. This paper is based on the RGB-D Odometry method to convert RGB-D image pairs into pointcloud and render visualization together. So the image quality of the RGB-D dataset directly affects the quality of the pointcloud. In the future, we will

first focus on how to efficiently select the known region information for complex RGB-D dataset to repair images and fill pointcloud data. Secondly, we will focus on the problem of pointcloud filling in overlapping regions, because this problem has been limited by 2D image restoration. In the existing pointcloud restoration technology, few technologies focus on detailed feature preservation and pointcloud edge information. We will focus on the edge information of images and the pointcloud, which can also be used for the accurate segmentation of images and pointclouds.

Author Contributions: Software, W.W., L.Z. and Z.L.; writing—original draft, H.Y.; writing—review & editing, Z.D. and J.H. All authors have read and agreed to the published version of the manuscript.

Funding: This manuscript was supported in part by Research Grant from MOE (Ministry of Education in China) Project of Humanities and Social Sciences (Grant number 18YJC880008), Project 'future school (infant education)' of National Center For Schooling Development Programme of China (Grant number CSDP18FC1201), Natural Science Foundation of Chongqing Municipality in China (Grant number cstc2018jcyjAX0835), and the Scientific Research Project Foundation of Chongqing University of Education (Grant numbers KY2018TZ01, 19GSKP01 and KY201903A).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Alexandru Telea. An Image Inpainting Technique Based on the Fast Marching Method. *J. Graph. Tools* **2004**, *9*, 23–34. [[CrossRef](#)]
2. Criminisi, A.; Perez, P.; Toyama, K. Region Filling and Object Removal by Exemplar-Based Image Inpainting. *IEEE Trans. Image Process.* **2004**, *13*, 1200–1212. [[CrossRef](#)] [[PubMed](#)]
3. Bertalmio, M.; Sapiro, G.; Caselles, V.; Ballester, C. Image Inpainting. In Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, New Orleans, LA, USA, 23–28 July 2000.
4. Telea, A.; Sapiro, G. Nontexture Inpainting by Curvature-Driven Diffusions. *J. Vis. Commun. Image Represent.* **2001**, *12*, 436–449.
5. Shen, C.J. Mathematical models of local non-texture inpaintings. *Siam J. Appl. Math.* **2002**, *62*, 1019–1043. [[CrossRef](#)]
6. Shen, J.; Kang, S.H.; Chan, T.F. Euler's Elastica and Curvature-Based Inpainting. *Siam J. Appl. Math.* **2003**, *63*, 564–592. [[CrossRef](#)]
7. Tsai, A.; Yezzi, A.; Willsky, A.S. Curve evolution implementation of the Mumford-Shah functional for image segmentation, denoising, interpolation, and magnification. *IEEE Trans. Image Process.* **2001**, *10*, 1169–1186. [[CrossRef](#)] [[PubMed](#)]
8. Esedoglu, S. Digital Inpainting Based On The Mumford-Shah-Euler Image Model. *Eur. J. Appl. Math.* **2003**, *13*, 353–370. [[CrossRef](#)]
9. Luo, K.; Li, D.; Feng, Y.; Zhang, M. Depth-aided inpainting for disocclusion restoration of multi-view images using depth-image-based rendering. *J. Zhejiang Univ. Sci. A* **2009**, *10*, 1738–1749. [[CrossRef](#)]
10. Xu, Z.; Sun, J. Image inpainting by patch propagation using patch sparsity. *IEEE Trans. Image Process.* **2010**, *19*, 1153–1165. [[PubMed](#)]
11. Anamandra, S.H.; Chandrasekaran, V. Exemplar-based color image inpainting using a simple and effective gradient function. In Proceedings of the 2010 International Conference on Image Processing, Computer Vision, & Pattern Recognition, Las Vegas, NV, USA, 12–15 July 2010.
12. Huang, J.B.; Kang, S.B.; Ahuja, N.; Kopf, J. Image completion using planar structure guidance. *ACM Trans. Graph.* **2014**, *33*, 1–10. [[CrossRef](#)]
13. Nie, Y.; Hou, J.; Han, X.; Niener, M. RfD-Net: Point Scene Understanding by Semantic Instance Reconstruction. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Virtual, 19–25 June 2021.
14. Steinbrücker, F.; Sturm, J.; Cremers, D. Real-time visual odometry from dense RGB-D images. In Proceedings of the IEEE International Conference on Computer Vision Workshops, Barcelona, Spain, 6–13 November 2011.
15. Park, J.; Zhou, Q.; Koltun, V. Colored pointcloud registration revisited. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017.
16. Tu, S.; Dong, L.; Yang, H.; Huang, Y. Complex variable moving Kriging interpolation for boundary meshless method. *Eng. Anal. Bound. Elem.* **2016**, *65*, 72–78. [[CrossRef](#)]
17. Quinsat, Y. Filling holes in digitized pointcloud using a morphing-based approach to preserve volume characteristics. *Int. J. Adv. Manuf. Technol.* **2015**, *81*, 411–421. [[CrossRef](#)]
18. Centin, M.; Pezzotti, N.; Signoroni, A. Poisson-driven seamless completion of triangular meshes. *Comput. Aided Geom. Des.* **2015**, *35*, 42–55. [[CrossRef](#)]
19. Li, S.; Yao, Y.; Tian, F.; Long, Q. Reconstructing Thin Structures of Manifold Surfaces by Integrating Spatial Curves. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018.

20. Yuan, W.; Khot, T.; Held, D.; Mertz, C.; Hebert, M. PCN: Point completion network. In Proceedings of the 2018 International Conference on 3D Vision, Verona, Italy, 5–8 September 2018; pp. 728–737.
21. Huang, Z.; Yu, Y.; Xu, J.; Ni, F.; Le, X. PF-Net: Point fractal network for 3d pointcloud completion. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Virtual, 14–19 June 2020.
22. Qi, C.R.; Su, H.; Mo, K.; Guibas, L.J. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Hawaii, HI, USA, 21–26 July 2017.
23. Qi, C.R.; Yi, L.; Su, H.; Guibas, L.J. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *arXiv* **2017** arXiv:1706.02413.
24. Zhang, Y.; Funkhouser, T. Deep depth completion of a single rgb-d image. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018.
25. Jeon, J.; Lee, S. Reconstruction-based pairwise depth dataset for depth image enhancement using CNN. In Proceedings of the European Conference on Computer Vision, Munich, Germany, 8–14 September 2018.
26. Xiao, J.; Ying, W. A Semi-fragile Watermarking Tolerant of Laplacian Sharpening. In Proceedings of the IEEE International Conference on Computer Science & Software Engineering, Wuhan, China, 12–14 December 2008.
27. Roosmalen, V. Restoration of Archived Film and Video. Ph.D. Thesis, Delft University of Technology, Delft, The Netherlands, 1999.