3DFill: Reference-guided Image Inpainting by Self-supervised 3D Image Alignment

Liang Zhao\textsuperscript{1} Xinyuan Zhao\textsuperscript{2} Hailong Ma\textsuperscript{1} Xinyu Zhang\textsuperscript{1} Long Zeng\textsuperscript{1}

\textsuperscript{1}Tsinghua University \hspace{1cm} \textsuperscript{2}Huawei Technologies

\{zhao-l20, mh120, zhangxy20\}@mails.tsinghua.edu.cn zhaoxinyuan1@huawei.com zenglong@sz.tsinghua.edu.cn

\begin{figure}
\centering
\begin{tabular}{cccc}
(a) Target & (b) Reference & (c) Before alignment & (d) After alignment & (e) Result \\
\end{tabular}
\caption{Results of our reference-guided inpainting algorithm. Our method uses a reference image as prior information to inpaint the masked target image and can solve cases with large holes or complex scenes.}
\end{figure}

Abstract

Most existing image inpainting algorithms are based on a single view, struggling with large holes or the holes containing complicated scenes. Some reference-guided algorithms fill the hole by referring to another viewpoint image and use 2D image alignment. Due to the camera imaging process, simple 2D transformation is difficult to achieve a satisfactory result. In this paper, we propose 3DFill, a simple and efficient method for reference-guided image inpainting. Given a target image with arbitrary hole regions and a reference image from another viewpoint, the 3DFill first aligns the two images by a two-stage method: 3D projection + 2D transformation, which has better results than 2D image alignment. The 3D projection is an overall alignment between images and the 2D transformation is a local alignment focused on the hole region. The entire process of image alignment is self-supervised. We then fill the hole in the target image with the contents of the aligned image. Finally, we use a conditional generation network to refine the filled image to obtain the inpainting result. 3DFill achieves state-of-the-art performance on image inpainting across a variety of wide view shifts and has a faster inference speed than other inpainting models.

1. Introduction

Image inpainting aims to restore missing pixels within a hole region in an image while making the restored image visually realistic. The hole region could contain occluded objects or corrupted regions of the image. Most of the existing algorithms focus on filling holes with the background information of a single image, which can be divided into the traditional algorithm and deep learning algorithm. The traditional algorithm implements hole region filling based on
pixel diffusion around the hole region [2] or similar pixel matching [3]. Deep learning algorithms based on neural network [12, 23, 25, 37] leverage knowledge learned from large-scale training data to fill in holes. Although these algorithms have achieved outstanding inpainting results in some cases, due to the ill-posed nature [42] of the task itself, it is difficult to solve the case with large hole or the hole region containing complex scenes.

The aforementioned difficulties can be addressed by reference-guided image inpainting [24] which has a second image from another viewpoint of the same scene. The second image which we called the reference image exposes the content that can be used to fill in the hole of the image which we called the target image. Although the reference image is easily obtained by moving the camera to a different viewpoint to expose the background or collecting from the Internet [36], reference-guided image inpainting is less explored due to the challenge of image alignment between the target image and reference image. Most of the existing reference-guided algorithms use 2D transforms, such as affine transformation and homography transformation [17,46], to achieve image alignment. Such methods simplify the problem of aligning images taken by cameras with a different viewpoint, resulting in poor effect. Although the captured image is 2D, the camera imaging process is a 3D to 2D mapping, which prevents simple 2D transformations from achieving the desired alignment.

To address these issues, we propose 3DFill, an image alignment scheme based on two-stage alignment: 3D projection + 2D transformation, and use the aligned image to complete the task of image inpainting. First, we use 3D projection to achieve the overall alignment of the two images. We reverse the camera imaging process to recover the 3D scene corresponding to the 2D target image. To implement it, we estimate the depth map of the target image and use the camera intrinsic parameter to calculate the corresponding 3D scene coordinates for each pixel. After recovering the 3D scene, we estimate the 6-DOF transformation matrix between the target image and the reference one, which help us to align the 3D scene from the target viewpoint to the reference one. We then use the forward process of camera imaging to project the 3D scene onto the 2D image at the reference viewpoint. The coarse-aligned image can be got by the differentiable bilinear sampling method [44]. Second, to improve the alignment accuracy of the hole region, we do a further fine 2D alignment between the coarse-aligned image and the target image. After getting the fine-aligned image, we fill the hole region of the target image with the contents of the fine-aligned image. Finally, for a realistic and natural filling effect, we use a simple conditional generation network (CGAN) [27] to refine the filled image. In our image alignment training process, we do not need the label of the transformation matrix between the image pairs and only use the image similarity after alignment as the loss function. Therefore this process is self-supervised training. The image alignment model and image refinement model of 3DFill are lightweight which makes the entire algorithm efficient and fast.

In summary, the main contributions of our method are:

- We propose 3DFill, a reference-guided image inpainting algorithm based on 3D image alignment.
- We propose a simple, efficient and self-supervised module to complete the image alignment task, which achieves great alignment effect.
- Experiments on the datasets indicate that our 3DFill is effective to generate realistic contents for image inpainting.

2. Related Work

**Image inpainting.** Inpainting methods can be divided into two categories: traditional methods and deep learning methods.

*Traditional methods:* Some traditional methods [2,5,16] use the technique of pixel diffusion to generate hole regions. However, when the hole area becomes larger, the restored effect of this kind of method will become blurred. Other methods [3,8,9] calculate the similarity of different regions and fill the hole regions with the most similar patches from non-hole regions. Although these methods can obtain high-quality texture, they will make the structure and semantic information of the filled image unreasonable.

*Deep learning methods:* Most of the deep learning methods use neural network to generate inpainted images. They can be divided into those with and without prior information. The input of the method without prior information [20,21,34,43] is only the image to be inpainted and the inpainting result is generated based on the complex model structure and large-scale training data. When the area of the missing region becomes larger and larger, the method without any prior information will fall into ill-posed nature, which leads to the poor and unreasonable restoration effect. To solve the problem of insufficient background information, more and more methods begin to leverage the knowledge learned from prior information, such as edges [23], low-frequency structures [29], segmentation masks [33], and reference images [41, 46]. Reference image contains the most complete structure and the highest quality textures compared to other prior information. Our method can be considered as a reference-guided image inpainting method.

**Image alignment.** Image alignment is the task of unifying different images into the same coordinate system. Usually, the image alignment task first needs to estimate the transformation matrix of the two images, that is, image
registration. Then warp one image to the another image through the transformation matrix. Based on different image registration methods, image alignment algorithms can be divided into 2D algorithms and 3D algorithms.

2D: Image 2D transformation can be divided into rigid body transformation, affine transformation, and homography transformation. Traditional methods utilize handcrafted descriptors [4, 22, 30] or learned local features [6, 7, 10] to estimate the homography matrix. Deep learning methods directly estimate the relative pose by an end-to-end network, such as OANet [39]. According to the principle of camera imaging, images from different viewpoints involve the transformation of the 3D scene and the projection from 3D to 2D. The transformation of the 3D scene includes 6 DOF, and the projection depends on the intrinsic parameters of the camera including 5 DOF, a total 11 DOF. A homography transformation only has 8 DOF which is difficult to achieve the desired image alignment. TransFill [46] designs a multi-homography strategy to realize 2D transformation, however, it is still not a good substitute for 3D transformation.

3D: The most classical 3D transformation algorithm is SfM [32], a machine learning algorithm. Although SfM achieves outstanding performance, it is difficult to achieve a fast registration speed due to the complexity of its calculation. Deep learning for directly predicting the 3D transformation, however, it is still not a good substitute for 3D transformation.

Reference-guided image inpainting. At present, there is not much research on image inpainting based on reference images. Most of these [17, 35] use vanilla homography matrix to complete 2D image alignment before performing image inpainting tasks. Still, others use SfM [1] or optimized homography [41, 46] matrices to achieve image registration, they are based on various assumptions, such as plane-based [26], and use sophisticated geometric algorithms to fit the image distortion caused by camera viewpoint change. These methods are not only computationally complex but also difficult to obtain fully aligned images, especially when the camera viewpoint changes largely. To achieve plausible image inpainting results in this situation, these methods often have to design a complex refining module, such as CST [46], which further complicates the model.

3. Method

Fig. 2 shows an overview of 3DFill. We are given an original image \(I_o \in \mathbb{R}^{W \times H \times 3}\), an associated mask \(M \in \mathbb{R}^{W \times H \times 1}\), and a reference image \(I_r \in \mathbb{R}^{W \times H \times 3}\). Note that \(M\) indicates the hole regions with value 0 and elsewhere with 1. The original image does not contain the hole region, and the target image containing the hole region is obtained by \(I_t = M \odot I_o\). Our method is divided into two steps, which correspond to two modules, namely, the image alignment (IA) module and the image refinement (IR) module. We first use the IA module, which has a two-stage alignment, to complete the image alignment task. The 3D module uses \(I_t\) and \(I_r\) as inputs and outputs the coarse-aligned image \(I_{ coarse}\). The 2D module uses \(I_t\), \(I_{ coarse}\) and \(M\) as inputs and outputs the fine-aligned image \(I_{ fine}\). We then fill the contents of the aligned image to the hole area of the target image to obtain the \(I_{ fill}\). After obtaining the \(I_{ fill}\), we use the IR module to implement the image refinement and output the inpainting result \(I_{ inpaint}\).

3.1. Image Alignment Module

3D Module. To achieve the ideal alignment effect, our algorithm first uses the 3D module for coarse alignment and the following three data are required: intrinsic parameter of the camera, depth map corresponding to the input image, and 3D transformation matrix between the target image and the reference image. In our 3D module, we use fixed camera intrinsic parameters instead of ground-truth camera intrinsic parameters because we find that our algorithm is robust to camera intrinsic parameters in Sec. 4.4. We use DPT [28], a state-of-the-art depth estimation model, to help us obtain relatively accurate depth map. This avoids the difficulty of

![Figure 2. An overview of 3DFill. First, we use the image alignment (IA) module to align the input target image with the reference image. The DPT [28] is used to estimate the depth map of the input image to achieve 3D projection. The Pose3DNet and Pose2DNet are used to estimate the 3D and 2D pose transformations between the input images, respectively. We fill the contents of the aligned image into the hole of the target image, and then use InpaintNet to refine the image by image refinement (IR) module.](image-url)
self-supervised training depth estimation models, allowing us to focus more on the accuracy of 3D projection. The 3D transformation matrix can be estimated by the Pose3DNet, which is a simple encoder model.

Fig. 3 shows the process of implementing 3D alignment. Step 1: we reverse the projection process according to the camera imaging principle to obtain the 3D scene. This process achieves a 2D to 3D projection with the help of the depth map $Z_t$ and camera’s intrinsic parameters $K$.

Step 2: we convert the scene coordinates from the target viewpoint to the reference viewpoint by 3D transformation. 3D transformation contains 6 degrees of freedom so we use Pose3DNet to predict a vector with 6 elements corresponding to 3 translation variables $t_x, t_y, t_z$ and 3 rotation variables $\alpha, \beta, \gamma$. We use the above variables to obtain the transformation matrix $T_{t\rightarrow r}$.

Step 3: we use the forward process of camera imaging to project the 3D scene onto the 2D image at the reference viewpoint.

So far, we can get the coordinates $P_r(u_r, v_r)$ of each pixel $P_t(u_t, v_t)$ on the target image projected on the reference image. The above projection process can be simplified by:

$$P_r = KT_{t\rightarrow r}Z_tK^{-1}P_t$$ (1)

$P_r$ is a decimal value and we use a differentiable bilinear sampling method [44] to get the coordinate of each pixel in the coarse-aligned image. Although the 3D transformation matrix we use is from the target view to the reference view, the actual effect is that we warp the reference image to the target view and get the coarse-aligned image $I_{coarse}$. We use the $L1$ loss between $I_{coarse}$ and $I_o$ to train the Pose3DNet:

$$L_{3D} = \|M_{valid} \odot (I_{coarse} - I_o)\|_1$$ (2)

where $M_{valid}$ indicates the area of the valid projection point with value 1 and elsewhere with 0. The areas where the target image and reference image do not overlap are replaced by black in $I_{coarse}$, as shown in Fig. 3.

Figure 4. The effect of alignment.

**2D Module.** Our 3D projection algorithm completely simulates the process of taking pictures with a camera and can achieve the right alignment when the camera is an ideal camera. However, the real camera is not ideal, and there may be interference such as lens distortion. Therefore, our 3D algorithm can achieve great alignment but sometimes is not fully aligned. To solve this problem, we add a step of 2D alignment after 3D alignment.

As shown in Fig. 4, the reference image and target image are coplanar after 3D alignment but still has some deviations. Most of these deviations are translations and rotations in the 2D image coordinate system, and a few deviations involve size differences, so we use Pose2DNet to achieve a scaled Euclidean transformation. The output of Pose2DNet is a vector with 4 elements corresponding to a 2D rotation angle $\theta$, two translation variables $t_x, t_y$, and a scale variable $s$, the $I_{fine}$ can be obtained by:

$$\begin{bmatrix}
    u_{fine} \\
    v_{fine} \\
    1
\end{bmatrix} = \begin{bmatrix}
    s \cos \theta & -s \sin \theta & t_x \\
    s \sin \theta & s \cos \theta & t_y \\
    0 & 0 & 1
\end{bmatrix} \times \begin{bmatrix}
    u_{coarse} \\
    v_{coarse} \\
    1
\end{bmatrix}$$ (3)

where $u_{fine}$ and $v_{fine}$ indicate the pixel coordinate of $I_{fine}$, $u_{coarse}$ and $v_{coarse}$ indicate the pixel coordinate of $I_{coarse}$.

We use the $L2$ loss between $I_{coarse}$ and $I_{fine}$ to train the Pose2DNet:

$$L_{2D} = \|(1 - M_{big}) \odot M_{valid} \odot (I_{fine} - I_o)\|_1$$ (4)

which means that we only calculate the loss around the hole region and it forces the Pose2DNet to pay more attention to the alignment accuracy of the hole region.

All of our image alignment algorithms use self-supervised training, which means that our model can be generalized to various image datasets with reference images regardless of whether there are image pose labels.
ous scenarios. For each scene, we sampled 3 images with the RealEstate10K dataset [45], which contains a series of labeled video frames. For ease of processing, we resized each image to a resolution of 512 × 512. We generated the mask of the human outline from the COCO dataset [19]. We divide the test dataset into two categories based on the area of the mask: small-mask with the mask area accounting for 5% ~ 15% of the image and large-mask with the mask area accounting for 15% ~ 30% of the image.

3.2. Image Refinement Module

After obtaining the \( I_{\text{fine}} \), we fill it to the hole region of the target image to generate a filled image \( I_{\text{fill}} \):

\[
I_{\text{fill}} = I_t \odot M + I_{\text{fine}} \odot (1 - M)
\]

\( I_{\text{fill}} \) roughly restores the missing area on the target image, however, there are still problems such as slight deviation in spatial position and inconsistency in brightness, which makes the whole picture look incoherent.

Our Image Refinement (IR) module uses a conditional generation network (CGAN) to improve the above issues. This CGAN in our algorithm is called as InpaintNet and its structure is shown in Fig. 5. The inputs of InpaintNet are \( I_t \) and \( M \), the model extracts features of different levels of the input images and inputs these features as conditional information or prior information into different levels of the generation network. The generation network samples a latent vector \( z \) from a standard Gaussian distribution and modulates \( z \) with the condition information to generate an image. To improve the sharpness of the generated images \( I_g \), we use PatchGAN [47] as a discriminator for adversarial training. The loss function of the generation network is:

\[
\mathcal{L}_{\text{inpaint}} = \lambda_1 \| (I_g - I_o) \|_1 + \lambda_2 VGG(I_g, I_o) + \lambda_3 \mathcal{L}_{\text{adv}}
\]

Here, the VGG loss is the perceptual loss matching features of 5 layers of a pre-trained VGG19 [31], \( \mathcal{L}_{\text{adv}} \) is the adversarial loss calculated by PatchGAN. Finally, we fill the \( I_g \) into the hole region of the target image to get the final inpainting result \( I_{\text{inpaint}} \):

\[
I_{\text{inpaint}} = I_t \odot M + I_g \odot (1 - M)
\]

4. Experiments

Datasets. We trained and evaluated the 3DFill on the RealEstate10K dataset [45], which contains a series of videos from YouTube taken by moving cameras in various scenarios. For each scene, we sampled 3 images with a displacement of 25 frames apart, we call this as original dataset, which contains 202K scenes. We used the trained IA module to test each scene and got the valid map \( M_{\text{valid}} \). We found that when the valid area in \( M_{\text{valid}} \) is between 60% ~ 80%, the range of viewpoint changes in the scene was reasonable and acceptable, in other cases, the viewpoint changes were too large or too small. Excessive viewpoint changes result in too little overlap between images, which is not the case for reference-guided image inpainting method to deal with. Cases with slight viewpoint changes are too simple for training so we abandoned them, but experiments show that 3DFill can handle such cases. We used this as a standard to clean the original dataset and got the final dataset, which contains 33.5K scenes, the training dataset has 32.5K scenes and the test dataset has 1K scenes.

For every scene, we set the first image as the original image, the second image as the small-view reference image, and the third image as the large-view reference image. That means we had 65K image pairs for training. We divided the test dataset into the small-view dataset and large-view dataset, consisting of the small-view reference image and the large-view reference image, respectively. This division helps test the alignment ability of the model to varying degrees of viewpoint change.

For ease of processing, we resized each image to a resolution of 512 × 512. We generated the mask of the human outline from the COCO dataset [19]. We divide the test dataset into two categories based on the area of the mask: small-mask with the mask area accounting for 5% ~ 15% of the image and large-mask with the mask area accounting for 15% ~ 30% of the image.

Implementation Details. We applied the pre-trained DPT [28] to estimate the depth map of input images and its model parameters were frozen during training. In IA module, we first trained the Pose3DNet for 120 epochs and then trained Pose2DNet for 30 epochs. Both of these models were optimized by Adam [14] with \( \beta_1 = 0.9, \beta_2 = 0.999 \), and a learning rate of \( 1 \times 10^{-4} \). In IR module, we trained the InpaintNet and PatchGAN together for 50 epochs and their optimizer were Adam with \( \beta_1 = 0.9 \) and \( \beta_2 = 0.999 \). The learning rate of InpaintNet is \( 1 \times 10^{-3} \) and the learning rate of PatchGAN is \( 1 \times 10^{-4} \). As for the loss function, we set \( \lambda_1 = 1, \lambda_2 = 200, \) and \( \lambda_3 = 0.01 \) in Eq. (6).

Compared Methods. We choose several representative image inpainting algorithms with or without prior information for comparison. We regard the video inpainting algorithm as a kind of image inpainting algorithm that takes adjacent frames as reference images.

FGVC [11]: Flow-edge Guided Video Completion is a video completion algorithm which first extracts and completes motion edges, and then uses them to guide piecewise-smooth flow completion with sharp edges.

STTN [38]: Spatial-Temporal Transformer Network fills missing regions in all input video frames by the proposed
Figure 6. Comparison with baselines on RealEstate10K. For better visualization, please zoom in to see the details.

4.1. Qualitative Comparison

Fig. 6 shows the qualitative comparison results on RealEstate10K. STTN and OPN can restore the approximate structure with the help of a small-view reference image. However, these video inpainting algorithms are based on simple geometric transformations to align adjacent frames, as the viewpoint of the reference image changes more and more violently, it is more difficult to extract meaningful geometric information and resulting in blurred and confusing images. OPN’s results are slightly better than STTN because OPN can perform image inpainting with fewer reference images, whereas STTN relies on coherent multiple adjacent frames. We found that when the scenario is relatively simple, MAT and LaMa can achieve good
Table 1. Quantitative comparisons on the RealEstate10K dataset.

| Model     | Small-view | Large-view | Small-view | Large-view |
|-----------|------------|------------|------------|------------|
|           | PSNR       | SSIM       | LPIPS      | PSNR       | SSIM       | LPIPS      | PSNR       | SSIM       | LPIPS      |
| FGVC      | 26.51      | 0.9475     | 0.0465     | 26.90      | 0.9468     | 0.0437     | 21.51      | 0.8792     | 0.1071     |
| STTN      | 29.63      | 0.9628     | 0.0376     | 29.49      | 0.9625     | 0.0380     | 24.42      | 0.9113     | 0.0914     |
| OPN       | 29.62      | 0.9630     | 0.0312     | 29.52      | 0.9616     | 0.0319     | 24.12      | 0.9104     | 0.0775     |
| EdgeConnect | 28.02  | 0.9570     | 0.0424     | 28.02      | 0.9570     | 0.0424     | 23.23      | 0.9058     | 0.0927     |
| MAT       | 29.14      | 0.9636     | 0.0290     | 29.14      | 0.9636     | 0.0290     | 23.77      | 0.9132     | 0.0708     |
| LaMa      | 30.46      | 0.9685     | 0.0259     | 30.46      | 0.9685     | 0.0259     | 25.33      | 0.9263     | 0.0639     |
| Ours      | 34.97      | 0.9773     | 0.0146     | 32.70      | 0.9690     | 0.0227     | 29.60      | 0.9404     | 0.0388     |

Table 2. Quantitative comparison of inference speed and model size.

| Model     | Speed/FPS | Params/M |
|-----------|-----------|----------|
| FGVC      | 0.50      | 5.26     |
| STTN      | 1.62      | 16.57    |
| OPN       | 1.73      | 3.42     |
| EdgeConnect | 3.16  | 67.11    |
| MAT       | 4.52      | 61.56    |
| LaMa      | 3.76      | 50.98    |
| Ours      | 5.46      | 42.11    |

4.2. Quantitative Comparison

Tab. 1 shows the quantitative comparison on the RealEstate10K dataset. We divided the test dataset into 4 types according to the mask size (Small-mask and Large-mask) and viewpoint variation of the reference image (Small-view and Large-view). For the single-image inpainting algorithms—EdgeConnect, MAT and LaMa, the results are the same in Small-view and Large-view. For the evaluation metrics, we use the PSNR, SSIM and LPIPS [40] scores based on AlexNet [15]. The quantitative results showed that FGVC failed to restore the target images. Although STTN and OPN achieved better results than FGVC, they still can’t compete with the single-image SOTA algorithm. MAT and LaMa can get better results in the case of Small-mask. Our 3DFill outperforms existing methods under various situations.

Considering the practicality of the algorithm, we compare the inference speed and model size of the above methods, the results are shown in Tab. 2. Although video inpainting models have a smaller number of parameters, their inference time is slower because they require multiple cycles to compute the features of all adjacent frames. Our model has fewer parameters than single-image inpainting algorithms and has the fastest inference speed of more than 5 FPS. This shows that our method does not depend on the complex design of the neural network, but relies on high-precision simulation of geometric relationship between images to achieve high-quality image inpainting results, which is more consistent with the laws of physics.

4.3. Ablation Study

To evaluate the effects of modules in our method, we design the following ablations (IA module consists of 3D alignment module and 2D alignment module):

1. Removing the IA module (w/o IA);
2. Removing the 3D alignment module (w/o 3D);
3. Removing the 2D alignment module (w/o 2D);
4. Removing the IR module (w/o IR);

As shown in Tab. 3 and Fig. 7, the IA module is the core of our method, without which we can hardly achieve the desired inpainting results. In the IA module, 3D alignment plays a major role because it roughly aligns two images. 2D alignment is an auxiliary alignment. The viewpoint change of the reference image has a significant impact on both 3D and 2D alignment. The IR module has little effect on image inpainting for small-view. However, when the viewpoint of
the reference image becomes larger, the IA module alone cannot achieve the ideal alignment effect. In this case, the IR module plays an important role in refining the images.

4.4. Robustness Study

3DFill uses fixed camera intrinsic parameters $K$ to achieve 3D projection because we can’t get ground-truth camera parameters in most cases. The camera intrinsic parameter contains 5 variables: the focal length $f$, the pixel length $d_x$ and $d_y$, the pixel coordinates of the origin of the image coordinate system $u_0$ and $v_0$. The parameter matrix $K$ is:

$$
K = \begin{bmatrix}
    f & 0 & u_0 \\
    0 & \frac{f}{d_x} & v_0 \\
    0 & 0 & 1
\end{bmatrix}
$$

The origin of the image coordinate system is in the center of the image so we can set the $u_0$ and $v_0$ be half the resolution of the image. Generally, the pixel lengths in the $x$ and $y$ directions are the same, so $d_x$ and $d_y$ can be standardized to take pixels as units, and the proportional relationship between pixels and the real length is transferred to $f$. So the only important variable is the focal length $f$. In this paper we set the focal length of all images to 750 pixels which is a common number. We test the robustness of 3DFill to focal length as shown in Tab. 4. The experimental results show that different focal lengths have little effect on 3DFill. We believe that the robustness of 3DFill to the focal length is

5. Conclusion

We propose 3DFill—a novel reference-guided image inpainting method. To maximize the use of information from the reference image, 3DFill utilizes a two-stage image alignment module to align the reference image with high accuracy and utilizes a simple image refinement module to further refine the alignment result. The inpainting results outperform state-of-the-art single-image and reference-guided image inpainting methods, especially when the hole is large or contains complicated scenes.

3DFill still has some limitations. First, the image alignment algorithm is not suited for image pairs with extreme viewpoint changes. Second, 3DFill uses the pre-trained DPT for depth estimation, which makes the accuracy of 3D projection dependent to some extent on the performance of DPT. Third, due to perspective, errors in depth estimation and lens distortion, the alignment effect of the image edge area is worse than that of the center area. When the hole region involve image edges and contain complex scenes, the image inpainting result is poor.
References

[1] Seung-Hwan Baek, Inchang Choi, and Min-H. Kim. Multiview image completion with space structure propagation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[2] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera. Filling-in by joint interpolation of vector fields and gray levels. IEEE Transactions on Image Processing, 10(8):1200–1211, 2001.

[3] C. Barnes. Patchmatch : A randomized correspondence algorithm for structural image editing. ACM Trans. Graphics, 28(3), 2009.

[4] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In Aleˇs Leonardis, Horst Bischof, and Axel Pinz, editors, Computer Vision – ECCV 2006, pages 404–417, 2006.

[5] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, pages 417–424, 2000.

[6] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. Dsac - differentiable ransac for camera localization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[7] Eric Brachmann and Carsten Rother. Neural-guided ransac: Learning where to sample model hypotheses. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019.

[8] A. Criminisi, P. Perez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. IEEE Transactions on Image Processing, 13(9):1200–1212, 2004.

[9] Soheil Darabi, Eli Shechtman, Connelly Barnes, Dan B. Goldman, and Pradeep Sen. Image melding: Combining inconsistent images using patch-based synthesis. ACM Trans. Graph., 31(4), 2012.

[10] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Toward geometric deep SLAM. CoRR, abs/1707.07410, 2017.

[11] Chen Gao, Ayush Saraf, Jia-Bin Huang, and Johannes Kopf. Flow-edge guided video completion. In European Conference on Computer Vision, 2020.

[12] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. ACM Trans. Graph., 36(4), 2017.

[13] Gagan Kanojia and Shammuganathan Raman. Mic-gan: Multi-view assisted image completion using conditional generative adversarial networks. In 2020 National Conference on Communications (NCC), pages 1–6, 2020.

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. Commun. ACM, 60(6):84–90, 2017.

[16] Levin, Zomet, and Weiss. Learning how to inpaint from global image statistics. In Proceedings Ninth IEEE International Conference on Computer Vision, pages 305–312 vol.1, 2003.

[17] Feiran Li, Gustavo Alfonso Garcia Ricardo, Jun Takamatsu, and Tsukasa Ogasawara. Multi-view inpainting for rgb-d sequence. In 2018 International Conference on 3D Vision (3DV), pages 464–473, 2018.

[18] Wenbo Li, Zhe Lin, Kun Zhou, Lu Qi, Yi Wang, and Jiaya Jia. Mat: Mask-aware transformer for large hole image inpainting. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022.

[19] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, Computer Vision – ECCV 2014, pages 740–755, 2014.

[20] Guolin Liu, Fitsum A. Reda, Kevin J. Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In Proceedings of the European Conference on Computer Vision (ECCV), 2018.

[21] Hongyu Liu, Ziyu Wan, Wei Huang, Yibing Song, Xintong Han, and Jing Liao. Pd-gan: Probabilistic diverse gan for image inpainting. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 9371–9381, 2021.

[22] D.G. Lowe. Object recognition from local scale-invariant features. In Proceedings of the Seventh IEEE International Conference on Computer Vision, pages 1150–1157 vol.2, 1999.

[23] Kamyar Nazeri, Eric Ng, Tony Joseph, Faisal Z. Qureshi, and Mehran Ebrahimi. Edgeconnect: Generative image inpainting with adversarial edge learning. CoRR, abs/1901.00212, 2019.

[24] Seoung Wug Oh, Sungho Lee, Joon-Young Lee, and Seon Joo Kim. Onion-peek networks for deep video completion. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019.

[25] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[26] Julien Philip and George Drettakis. Plane-based multi-view inpainting for image-based rendering in large scenes. In Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, 2018.

[27] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. arXiv e-prints, 2015.

[28] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 12179–12188, 2021.

[29] Yurui Ren, Xiaoming Yu, Ruonan Zhang, Thomas H. Li, Shan Liu, and Ge Li. Structureflow: Image inpainting
via structure-aware appearance flow. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[30] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011.

[31] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

[32] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3d. In *ACM SIGGRAPH 2006 Papers*, page 835–846, 2006.

[33] Yuhang Song, Chao Yang, Yeji Shen, Peng Wang, Qin Huang, and C.-C. Jay Kuo. Spg-net: Segmentation prediction and guidance network for image inpainting. *CoRR*, abs/1805.03356, 2018.

[34] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2149–2159, 2022.

[35] Theo Thonat, Eli Shechtman, Sylvain Paris, and George Drettakis. Multi-view inpainting for image-based scene editing and rendering. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 351–359, 2016.

[36] Oliver Whyte, Josef Sivic, and Andrew Zisserman. Get out of my picture! internet-based inpainting. In *BMVC*, 2009.

[37] Wei Xiong, Jiahui Yu, Zhe Lin, Jimei Yang, Xin Lu, Connelly Barnes, and Jiebo Luo. Foreground-aware image inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[38] Yanhong Zeng, Jianlong Fu, and Hongyang Chao. Learning joint spatial-temporal transformations for video inpainting. In *The Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[39] Jiahui Zhang, Dawei Sun, Zixin Luo, Anbang Yao, Lei Zhou, Tianwei Shen, Yurong Chen, Long Quan, and Hongen Liao. Learning two-view correspondences and geometry using order-aware network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[40] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[41] Yunhan Zhao, Connelly Barnes, Yuqian Zhou, Eli Shechtman, Sohrab Amirghodsi, and Charless C. Fowlkes. Geofill: Reference-based image inpainting of scenes with complex geometry. *CoRR*, abs/2201.08131, 2022.

[42] Chuanxia Zheng, Tat-Jen Cham, and Jianfei Cai. Pluralistic image completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[43] Chuanxia Zheng, Tat-Jen Cham, and Jianfei Cai. Pluralistic image completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[44] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[45] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *CoRR*, abs/1805.09817, 2018.

[46] Yuqian Zhou, Connelly Barnes, Eli Shechtman, and Sohrab Amirghodsi. Transfill: Reference-guided image inpainting by merging multiple color and spatial transformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2266–2276, 2021.

[47] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.
Appendices

A. 3D Alignment in Detail

In this section, we introduce the 3D alignment algorithm in detail. Fig. 3 shows the process of implementing 3D alignment.

Step 1: we reverse the projection process according to the camera imaging principle to obtain the 3D scene. The coordinates of the 3D scene corresponding to each pixel on the target image can be calculated by:

\[
\begin{bmatrix}
X_t \\
Y_t \\
Z_t \\
1
\end{bmatrix} =
Z_t \cdot
\begin{bmatrix}
\frac{f}{d_x} & 0 & u_0 \\
0 & \frac{f}{d_y} & v_0 \\
0 & 0 & 1
\end{bmatrix}^{-1}
\begin{bmatrix}
u_t \\
v_t \\
1
\end{bmatrix}
\]  

\( (8) \)

\( u_t, v_t \) indicates the pixel coordinates on the target image. \( X_t, Y_t, Z_t \) are the coordinates of the 3D scene from the target viewpoint and \( Z_t \) is the depth estimate of the DPT. The first matrix on the right side of the equation is the inverse of the camera intrinsic matrix, where \( f \) is the focal length, \( d_x \) and \( d_y \) indicate the pixel lengths in the \( x \) and \( y \) directions of the image respectively, \( u_0 \) and \( v_0 \) indicates the coordinates of the origin of the image coordinate system in the pixel coordinate system. Briefly speaking, this process achieves a 2D to 3D projection with the help of the depth map and camera’s intrinsic parameters.

Step 2: we convert the scene coordinates from the target viewpoint to the reference viewpoint by 3D transformation. 3D transformation contains 6 degrees of freedom so we use Pose3DNet to predict a vector with 6 elements corresponding to 3 translation variables \( t_x, t_y, t_z \) and 3 rotation variables (Euler angles) \( \alpha, \beta, \gamma \). We compose the translation variables into the vector \( t \) and convert three Euler angles to a rotation matrix \( R \). We can get the coordinates in the reference viewpoint by:

\[
\begin{bmatrix}
X_r \\
Y_r \\
Z_r \\
1
\end{bmatrix} =
\begin{bmatrix}
R & t \\
0^T & 1
\end{bmatrix}
\begin{bmatrix}
X_t \\
Y_t \\
Z_t \\
1
\end{bmatrix}
\]  

\( (9) \)

Step 3: we use the forward process of camera imaging to project the 3D scene onto the 2D image at the reference viewpoint by:

\[
\begin{bmatrix}
u_r \\
v_r \\
1
\end{bmatrix} =
\frac{1}{Z_r}
\begin{bmatrix}
\frac{f}{d_x} & 0 & u_0 \\
0 & \frac{f}{d_y} & v_0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
X_r \\
Y_r \\
Z_r
\end{bmatrix}
\]  

\( (10) \)

So far, we can get the coordinates \( P_r(u_r, v_r) \) of each pixel \( P_t(u_t, v_t) \) on the target image projected on the reference image. The above projection process can be simplified by:

\[
P_r = \frac{Z_r}{Z_t} KT_{t\rightarrow r} K^{-1} P_t
\]  

\( (11) \)

\( K \) is the intrinsic camera matrix, \( T_{t\rightarrow r} \) is the camera transformation matrix.

B. Model Architecture

In this section, we show the detailed architecture of the models used by 3DFill. Both the Pose3DNet and Pose2DNet are encoders as shown in Fig. 8, Fig. 9, and the InpaintNet is a CGAN as shown in Fig. 10.

C. Additional Visual Results

The alignment results of 3DFill are shown in Fig. 12. The additional qualitative comparisons are shown in Fig. 13.

D. Failure Cases

Most failure cases of 3DFill are caused by holes that contain complex scenes at the edge of the image. It is difficult to align the edge of the image as well as the center of the image due to perspective, errors in depth estimation and lens distortion, especially when some complex objects appears at the edge of the image. Under this condition, although the overall structure in the hole regions can be restored, structural dislocation may occur at the edge of some hole regions as shown in Fig. 11.

The more realistic camera intrinsic parameters considering lens distortion parameters and the more accurate depth estimation model can alleviate the above problems.
Figure 8. Model architecture of Pose3DNet. $D_t$ and $D_r$ are the depth maps of $I_t$ and $I_r$ estimated by DPT.

Figure 9. Model architecture of Pose2DNet.

Figure 10. Model architecture of InpaintNet.

Figure 11. Failure case. In this case, the viewpoint of edge region (such as the tables and chairs at the bottom left of the image) has changed too much due to perspective and lens distortion, resulting in poor alignment.
Figure 12. Image alignment results.
Figure 13. Additional qualitative comparisons. For better visualization, please zoom in to see the details.