Drag Your GAN

摘要

随着生成模型的兴起，人们对于图像生成和编辑的要求也越来越高，对图像内容的各方面都有灵活和精确的可控性要求。图像编辑领域中，DragGAN 提出了一种通过用户输入点对的方法，实现了较为精细的图像编辑，但是当用户输入的点对位于单调纹理处时，效果不佳。针对这一点，本文使用 Qt 框架复现了 DragGAN 的流程后，通过一个简单的点对位置更新的方法，让点对在开始编辑前移动到邻近区域纹理更丰富的位置来实现一个较好的图像编辑效果。通过进行人脸关键点从源图向目标图编辑的实验，本方法对图像编辑的有一定的改进。实验结果表明，点对初始的邻近区域纹理丰富程度对图像编辑的效果存在影响，选在纹理丰富区域的点对会有更好的编辑效果。这为改进 DragGAN 的编辑效果提供了新的思路，同时提供了一种简单的实现。

关键词： DragGAN；图像编辑；Qt

1 引言

随着深度学习的发展，计算机视觉领域也迅速发展。在图像生成领域，出现了许多生成模型如 VAE [4]、GAN [1] 等。在生成对抗网络（GAN）提出后，掀起了图像生成的热潮。

随后出现了大量基于 GAN 的模型，从图像编辑、风格迁移、合成图片等多个方向进行研究和发展。StyleGAN [3] 就是其中的一个模型。StyleGAN 中引入了风格的概念，将生成器的输入分解成风格和内容两部分，从而实现了对图像风格的细粒度控制，类似于风格迁移技术 [2]。此外 StyleGAN 还引入了一个映射网络 MappingNetwork，得到了一个将潜码空间映射到一个可控的中间潜码空间的概念。这个概念使得生成器的输入更易于调整，有助于实现更精细的控制。

在 StyleGAN 的基础上，DragGAN [5] 利用 StyleGAN 提出的中间潜码空间中图像的特征潜码在其中良好的可区分性，通过在中间潜码空间中迭代更新潜码从而实现图像的编辑，类似于风格迁移。DragGAN 主要完成了两步工作：运动监督和点追踪。在运动监督中，DragGAN 主要实现了对图像特征在潜码空间上的更新；而在点追踪中，由于运动监督更新之后，原来的点对中的操作点位置虽然更新了，但是更新的新位置并不明确，因此需要对比更新前后两种图像在操作点邻域的相似度来估计新点位置。

在单调纹理区域的编辑中，DragGAN 方法存在困难，导致编辑效果不尽如人意。在文献中，研究者建议用户输入点对时选择纹理更加丰富的区域，但并没有给出解决方案或是可能的尝试，因此仍然存在改进的空间。本文在 DragGAN 的基础上，针对 DragGAN 的操作点处于单调纹理区域时，点追踪会出现近乎停滞的现象，提出了一个在图像进行更新迭代之前，先寻找并靠经一个丰富纹理区域的改进思路和一种简单的实现方法。

2 相关工作

近年来，图像生成和编辑领域经历了显著的发展，涌现出多种方法，其中一些集中在通过用户输入点对实现更为精准的图像编辑。DragGAN 引入了一种基于点对的图像编辑方法，为用户提供了更加灵活和精细的编辑控制。也有许多其他类似的工作利用 GAN 或扩散模型进行可控图像合成。

本文尝试通过简单而有效的方法改进 DragGAN 在单调纹理区域的图像编辑效果。这种方法的灵感部分来自于以下论文：SSIM [7]，即结构相似性指标来测量图像相似性，它可以帮助找到图像中的结构和问题。以及使用局部二值模式 [6] 可以进行纹理分析，通过比较像素与其邻域像素的灰度值，可以有效捕捉图像的纹理信息。

3 本文方法

3.1 本文方法概述

DragGAN 的方法，主要是完成了图像编辑的工作。准备工作是利用预训练好的 StyleGAN 生成一张图片，并拿到它的潜码 ω。然后由用户在图片上打点，每两个点一组，每组中第一个点为操作点 p，第二个点为目标点 t。至此，准备工作完成。然后通过 DragGAN 中实现的动作监督和点追踪，实现图像的更新和操作点的位置，一直到操作点 p 移动到目标点 t 足够近的位置。

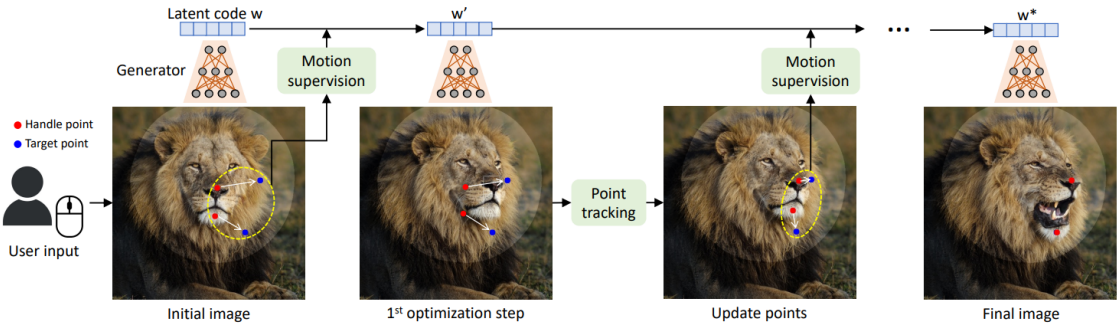

图 1. 方法示意图。图引自 [5]

3.2 动作监督

DragGAN 实现图像编辑的第一步便是动作监督。简单来说，动作监督就是把用户输入的点想着目标点进行移动。利用用户输入的操作点 p 和目标点 t 以及潜码 ω，代入损失函数(1)，通过梯度下降法实现对潜码的更新。

$$L = \sum_{i=0}^{n} \sum_{q_i \in \Omega_i(p_i, r_1)} ||F(q_i) - F(q_i + d_i)||_1 \tag{1}$$

以上公式中，F(q) 表示像素点 q 在特征图 F 中的特征值，p_i 表示第 i 个点对中的操作点 i，$\Omega_i(p_i, r_1)$ 中 r_1 是一个超参数，整体则表示取以 p 点为圆心，以 r_1 为半径取邻域，d_i 表示从 p_i 到 t_i 的方向向量，通过 $d_i = \frac{t_i - p_i}{||t_i - p_i||_2}$ 求得。

3.3 点追踪

DragGAN 图像编辑的第二步便是点追踪。简单来说，点追踪就是把移动后的操作点找出来。在第一步的动作监督实现对潜码 ω 的更新后，我们并不知道原来的操作点 p 移动到哪里去了，因此需要估计其位置。实现的思路是，在原来的操作点 p 附近找点，在其中取与原来的操作点 p 特征最接近的点作为新的操作点。具体实现，参考公式(2)。

$$p_i := \arg\min_{q_i \in \Omega_2(p_i, r_2)} ||F'(q_i) - f_i||_1 \tag{2}$$

其中 $\Omega_2(p_i, r_2)$ 中，r_2 是另一个超参数，整体表示以 p_i 为圆心，r_2 为半径的邻域，$F'(q_i)$ 表示像素点 q_i 在新的特征图 F' 的特征值。

4 复现细节

4.1 与已有开源代码对比

本次复现中，主要参考了 DasudaRunner 实现的 DragGAN 主要功能的代码，其代码仓库地址为https://github.com/DasudaRunner/DragGAN。主要借鉴了其对 StyleGAN 的封装代码，以及其实现 DragGAN 的基本思路。主要参考文件包含 StyleGAN 官方仓库中实现 StyleGAN 代码的 stylegan2_gan 文件夹中的所有内容，DasudaRunner 在 stylegan2_gan 文件夹中添加的实现了 StyleGAN 封装的 model.py 文件，最后是 DasudaRunner 在 backend.py 文件中引用 StyleGAN 封装实现的 DragGAN 的功能。

在 DragGAN 的实现部分，为了提供和 DragGAN 官方代码相同的功能，使用了 Qt 框架来实现了 DragGAN 功能的系统界面，而非 DasudaRunner 所使用的 dearpygui 框架。此外，为了能够方便的调整各种参数和切换潜码空间，添加了许多控制变量，使系统功能更完善。通过实现自定义的图片自适应组件，使得图片显示不受限与图片大小，而是与界面窗口相一致。

此外，为了实现图像反演功能，使用了 PTI 技术，使用了 PTI 作者的代码来实现图像反演，并为其实现了界面 UI。PTI 作者的代码仓库地址为https://github.com/danielroich/PTI。

为了实现论文中的实验，实验部分的代码并没有任何可以参考的资料，于是借助了 dlib 库实现人脸关键点的检测，根据论文中实验部分的描述完成了实验部分代码的编写。同时，将实验部分的功能加入到的系统界面上，便于进行实验。其他各个超参数的调整同样加入到了用户界面上，使系统功能调节更方便。

最后，在以上内容的基础上，添加了 Nearest 模块，在 DragGAN 的首步调整之前，对输入的操作点进行了一些调整，在提高图片编辑效果上有一定效果。具体实现则是将用户的操作点进行微调，使其在开始编辑之前移动到纹理更丰富的位置，便于后续的点追踪部分能够更好的捕捉到新操作点的位置。

4.2 实验环境搭建

本项目使用 Python 作为编程语言实现，使用的 Python 版本为 3.7，在 Anaconda3 提供的 Python 虚拟环境中完成实验环境的搭建。

本项目需要使用 Pytorch 框架来运行模型，使用的版本要求为 >=2.0.0，实际使用的版本为 2.0.1。对应使用的 CUDA 版本为 11.7。

本项目需要使用 StyleGAN 模型，因此需要配置好 StyleGAN 所需的环境。所需的 StyleGAN 代码来自 Nvidia 的 StyleGAN 的代码仓库。同时，为了能够使用预训练好的 StyleGAN，还需要准备 StyleGAN 所需的 *.pkl 模型参数文件。具体的下载，可以使用 DragGAN 官方目录下 scripts 文件夹下提供的 download_model.py 文件来实现，或是找到同目录下的 download_models.json 文件，其中有参数文件对应的详细地址，拿到地址后，可以手动下载。

为了能够顺利运行 StyleGAN 模型，需要安装 Visual Studio 2019，并修改在 torch_utils 文件夹下的 custom_ops.py 文件，将其中 _find_compiler_bindir() 函数中的 patterns 变量内的路径注释掉，并添加本地安装的 Visual Studio 2019 中所带的 MSVC 工具，具体的目录为原来的 patterns 变量的目录对应的本地目录。

由于使用了 QT 框架来实现 DragGAN 的界面，需要使用 python 安装 PySide6 库。由于需要使用 dlib 来实现人脸关键点检测，需要使用 python 安装 dlib 库。由于需要使用 ninja 来实现 StyleGAN 中依赖文件的构建，因此需要使用 python 安装 ninja 库。由于会用到 opencv 库对图像进行处理，需要使用 python 安装 opencv-python 库。

4.3 界面分析与使用说明

系统 UI 的实现，采用了 Python 中的 PySide6 依赖库，即 Qt6 框架来完成。

首先需要完成界面设计。本系统仅包含一个主界面，但主界面中包含了模型加载、图像生成参数、图像调整参数、实验参数和图像反演参数五个功能块的界面，此外还需要一个显示图像、用户输入点对以及图像更新的部分，因此界面中元素较多。整体界面如图2所示。

在 UI 的基本信息部分，可以选择使用 cpu 或 cuda 运行 DragGAN 的编辑过程，界面如图3所示。使用 cuda 即使用显卡运行。在 Device 下方，点击 Browse 按钮可以在计算机上选择 *.pkl 模型文件。当选过任意一个模型文件后，会记录最后一个选择的模型文件，点击 Recent 按钮会自动加载最后一次选择的模型文件。Embedding 部分用于实现图像反演时，选择反演模型对应的 *.pt 格式的 Embedding 文件。Latent 部分用于选择不同的生成 seed 值，seed 值的范围在 0 65535。如果启用 Random Seed 选项则不可手动编辑 seed 值，而是在每次点击 Generate 按钮后自动生成一个范围内的 seed 值。设置后 seed 值后，点击 Generate 即可在右侧显示生成的图片。图片成功显示后，点击 Save 按钮即可保存图片到源代码文件目录。

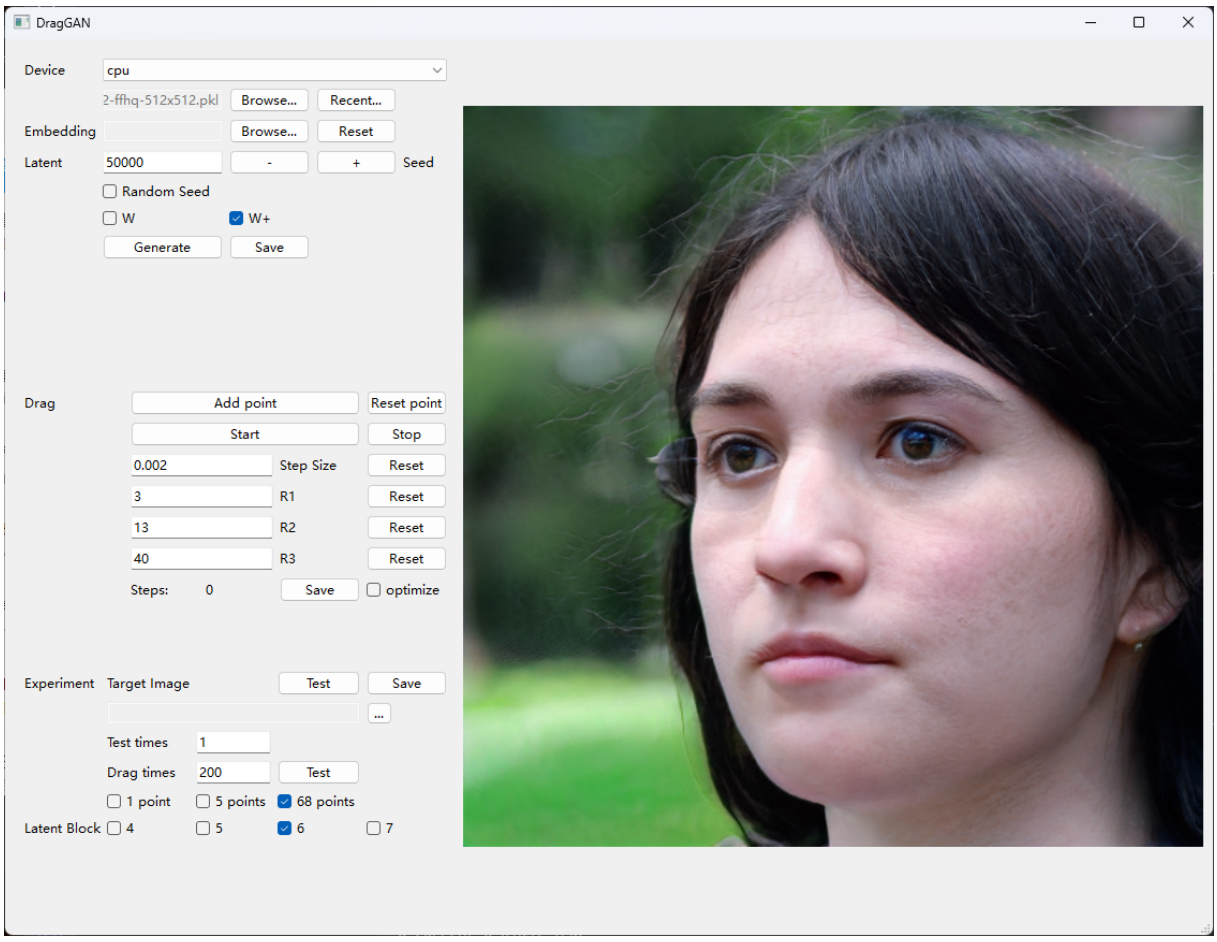

图 2. 系统整体界面

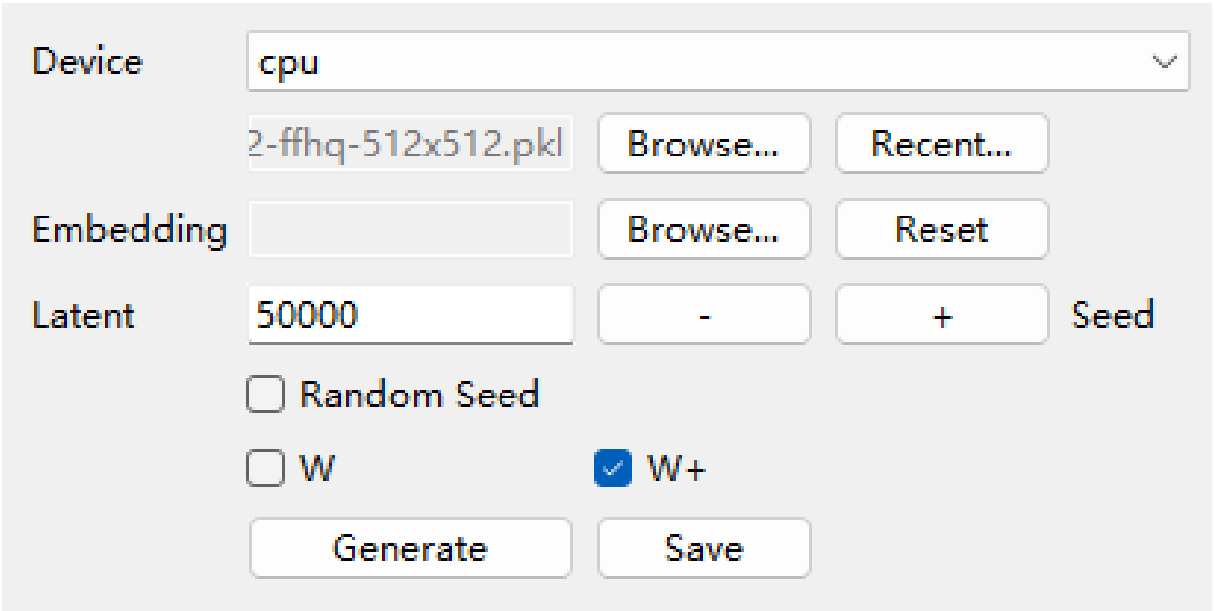

图 3. 基本信息部分界面

在 Drag 部分，点击 Add point 按钮后，可以在图像上点击操作点和目标点，界面如图4所示。点击 Reset point 按钮后，会清除之前添加的所有点。点击 Start 按钮即可开始编辑，编辑开始后直到操作点移动到目标点或用户点击 Stop 按钮后编辑结束。Step Size、R1、R2、R3

为用户可以手动编辑的超参数值，其后的 Reset 按钮会将对应的超参数值重新设置为默认值。

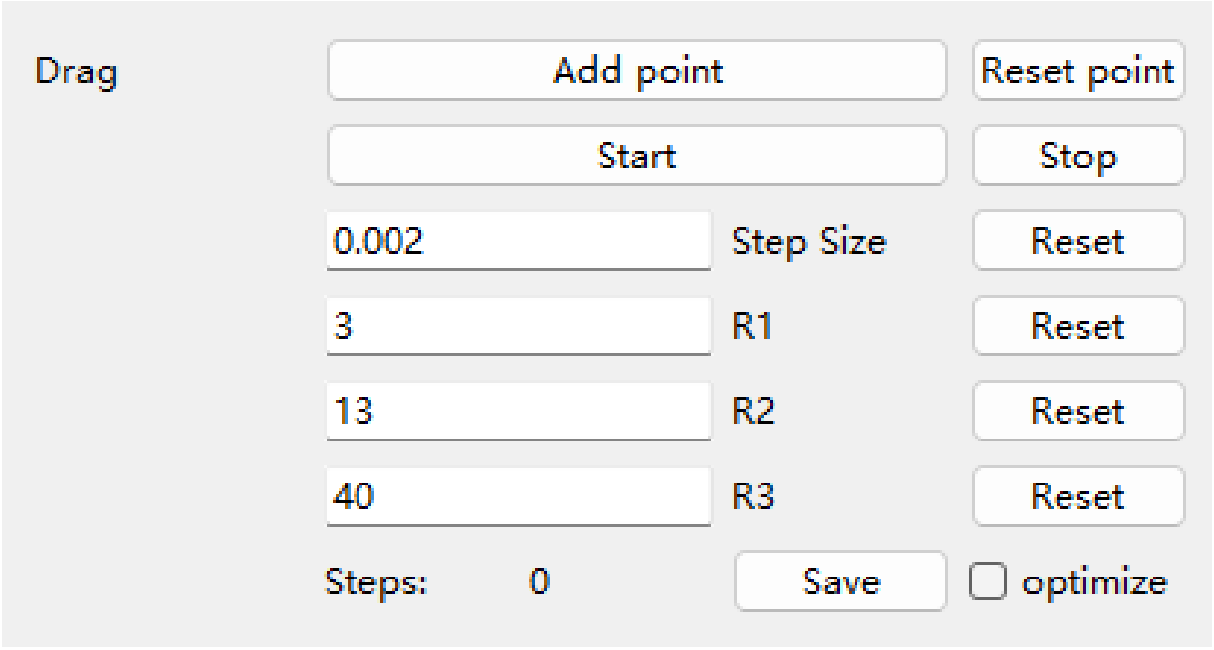

图 4. Drag 部分界面

Experiment 部分提供了单对图像测试和多对图像测试两个功能，界面如图5所示。Target Image 部分供用户选择目标图像。开始测试前，需要用户先生成图像在右侧显示部分，然后点击 Target Image 右侧的 Test 按钮即可开始单对图像的人脸关键点的实验。在 Test times 处可以设置多对图像测试的图像对数，并可以在 Drag times 处设置每一对图像的最大调整次数，设置好基本信息和 Test times、Drag times 两个值后，点击 Drag times 之后的 Test 按钮即可开始多对图像测试。两种测试都可以选择 1 个、5 个或 68 个人脸关键点进行测试。此外，还提供了设置潜码空间的不同块处开始进行训练的 Latent Block 值。

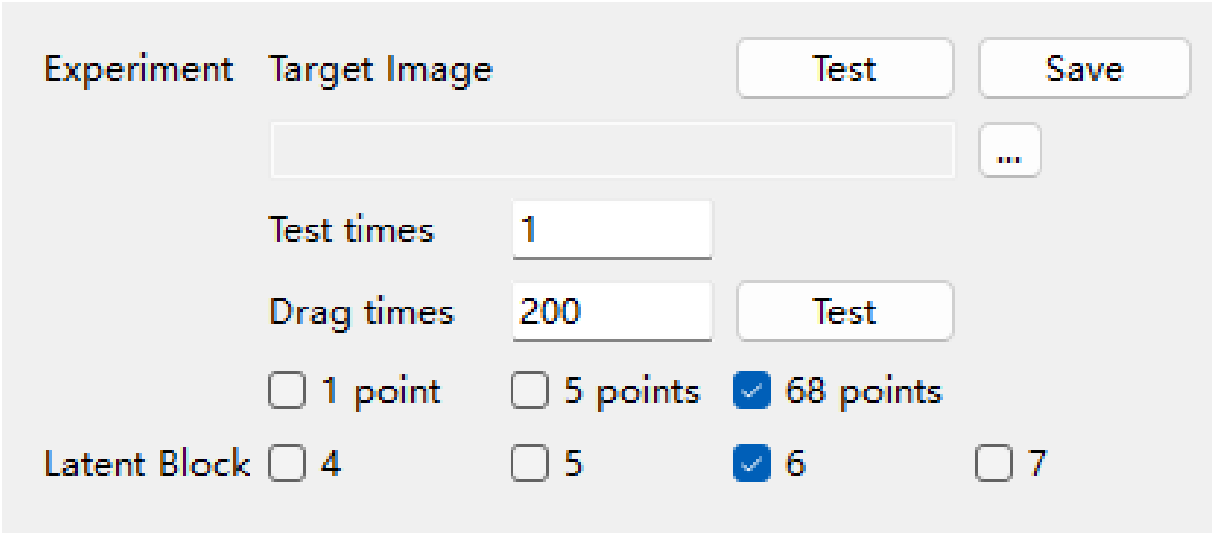

图 5. Experiment 部分界面

在实现 UI 的过程中，由于图片大小与界面大小可能不一致的原因，还需要考虑用户输入的位置与用户输入在图片的实际位置的缩放问题。为了解决这一问题，借助 Qt 提供的组

件，实现了图像自适应且绘制位置自适应组件 ImageWidget。其中绘制位置自适应由在 ImageWidget 组件中组合了另一个自定义组件 ImageLabel 完成。这两个组件主要通过记录图片实际大小和图片显示大小的比率来完成图像的自适应显示和绘制位置的自适应记录。

4.4 创新点

4.4.1 创新思路

由于 DragGAN 中，当用户选点在单调纹理处时，点追踪难以找到更新的位置。据此，新增的 Nearest 模块将操作点先向纹理丰富处移动一段，使其在点追踪时可以更好地借助周围纹理特征来实现点追踪。

4.4.2 思路实现

对于纹理丰富的考量，我将图片上有轮廓的区域认为是纹理较丰富的位置。具体的实现如下：

首先在操作点的一个 R3 邻域内取点。R3 是一个超参数。在 R3 邻域内，使用 canny 算子得到这一块区域中的所有轮廓。随后，在轮廓上依次取点。为使操作点移动的距离尽可能的小，对取得的每一个点，依次计算该点到操作点的距离，找到轮廓中所有的点中，距离操作点最近的点作为最近轮廓点。

考虑到点在轮廓上时可能导致编辑效果与预期不符，因此考虑在原操作点到最近轮廓点之间取一个离最近轮廓点更近的点作为新的操作点。距离最近轮廓点有多近，这个程度通过参数 rate 表示。当 rate 取 0.8 时，表示原操作点将移动从原操作点到最近轮廓点的距离间为 4/5 处的距离。至此，完成对一个点对的更新。

对于用户所选的每一个点对中的操作点，都在 DragGAN 的调整开始之前进行以上操作，将原操作点调整到一个纹理更丰富的点，从而实现在不改变编辑原意的前提下达到更好的编辑效果。

5 实验结果分析

5.1 人脸调整实验

5.1.1 实验设计

本实验的调整目标，首先是获得一张源图片和一张目标图片。然后分别从两张图片获得人脸关键点的坐标，并将两组人脸关键点一一对应组成人脸关键点调整点对。这个调整点对就作为用户输入的点对。然后调整的目标是让源图像向目标图像靠近，如图6所示。

在这里的人脸关键点提取中采用了三种方式，分别是：dlib 的 68 点人脸关键点检测、dlib 的 5 点人脸关键点检测和从 dlib 的 5 点人脸关键点检测的点取出鼻尖点作为单点人脸关键点。每一对图像的调整次数上限为 300 次，实验每次进行 100 对图像的调整，取其结果均值作为实验结果。最后我们使用 MeanDistance 指标和 FID 分数作为实验结果的指标。这里的 MeanDistance 指标表示调整结束后，整组点对的平均距离。

实验设置可以在 UI 界面中进行设置。如图7所示。

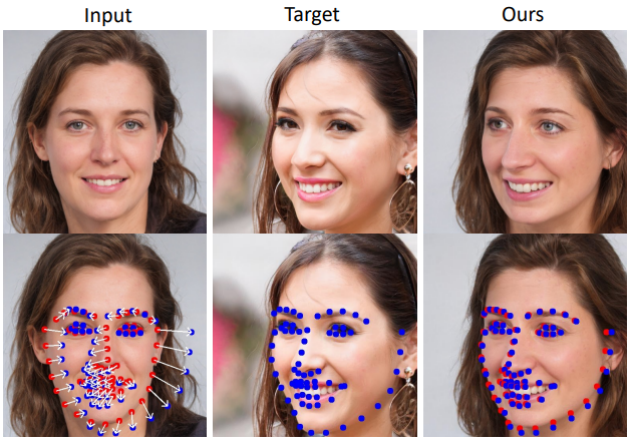

图 6. 人脸调整实验示意图，图引自 [5]

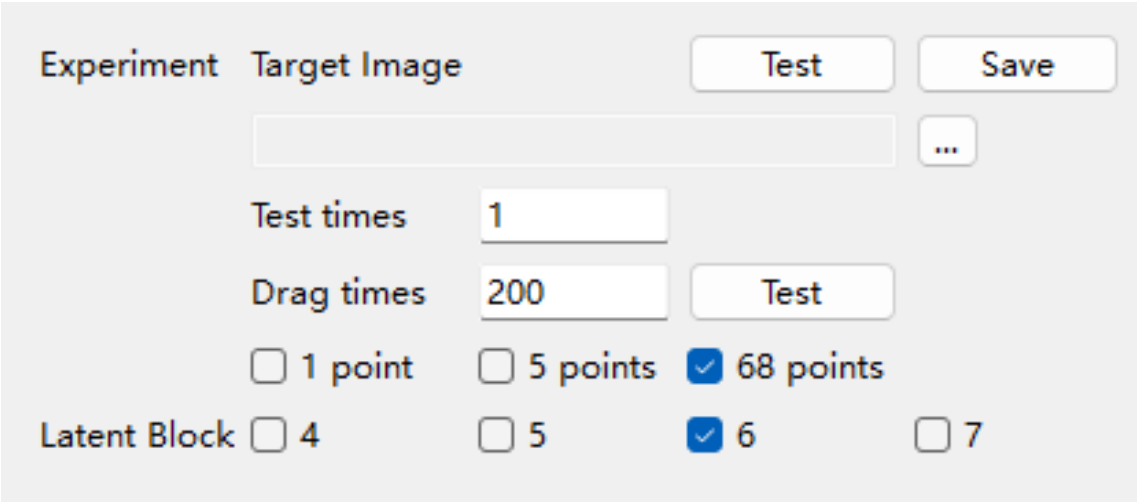

图 7. 实验部分 UI

5.1.2 实验结果

经过多次实验，结果如表1所示。其中 FID 分数是人脸关键点取 1 点时得到的结果。

Mean Distance	1 Point	5 Points	68 Points	FID
Ours	4.69	3.59	11.62	91.765
Ours+Nearest	4.81	3.28	16.16	58.627

表 1. 人脸关键点取不同数量，以及是否采用改进方法时的结果

5.1.3 结果分析

通过结果来看，虽然使用改进后 MeanDistance 不太稳定，有优化而有降低，但是从 FID 分数来看，采用改进方法的结果图像的编辑效果要更好。

5.2 消融实验

5.2.1 实验设计

消融实验主要包括：R3 的大小设置；通过 R3 设置不同值，找到最合适的 R3 的大小。

本实验中采用的实验方式仍然是人脸调整的实验方式，人脸关键点采取单点设置，这样运行得更快。本实验的评价指标则是 MeanDistance 指标。

5.2.2 实验结果

经过对 R3 设置不同取值，得到结果如表2所示。

Mean Distance	10	20	30	40	60
Ours+Nearest	4.72	4.23	4.71	4.18	4.69

表 2. R3 不同取值的影响

5.2.3 结果分析

从实验结果中可以看出，对 R3 取 40 时，效果是最好的。

6 总结与展望

通过本研究，我们成功地复现了 DragGAN 并在其基础上引入了一种简单而有效的点对位置更新方法，以改善在单调纹理区域的图像编辑效果。我们的实验结果表明，选择初始点对位置时考虑邻近区域纹理的丰富程度对于提高编辑质量有一定帮助。

在未来的工作中，我们可以进一步探索如何自动选择初始点对位置，以提高用户友好性，并考虑在不同数据集和场景下的通用性。此外，我们的方法虽然在人脸关键点编辑上取得了一定的成功，但其在其他图像编辑任务中，如光照迁移、细粒度调整等问题的适用性还有待深入研究。DragGAN 在其他方面，如掩码设置，都还有可以改进的空间，这些内容也有待深入研究。

综上所述，本研究为改进 DragGAN 的图像编辑效果提供了一种简单可行的方法，我们期待未来的工作能够进一步推动图像编辑技术的发展，以满足用户对于更高级、更精细编辑的需求。

参考文献

[1] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65, 2018.

[2] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *Journal of Vision*, 2015.

[3] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, and Timo Aila. Analyzing and improving the image quality of stylegan. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[4] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[5] Xingang Pan, Ayush Tewari, Thomas Leimkühler, Lingjie Liu, Abhimitra Meka, and Christian Theobalt. Drag your gan: Interactive point-based manipulation on the generative image manifold, 2023.

[6] Nasir Mahmood Rajpoot. Texture classification using discriminant wavelet packet subbands. In *The 2002 45th Midwest Symposium on Circuits and Systems, 2002. MWSCAS-2002.*, volume 3, pages III–III. IEEE, 2002.

[7] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.