数字图像处理项目报告

赖帅全 10225101417、路修远 10210740410、矫承达 10225101402

2024.7

一、 摘要

本论文介绍我们小组的项目成果网页 APP，允许用户在互联网浏览器中进行图片处理。其核心特性包括用户自由上传和下载图片、多种图片编辑功能、以及让图片变得可动的独特功能。通过这些功能，用户可以快速轻松地编辑自己的图片，从而实现个性化的效果。在本文中，将详细介绍项目成果的功能、技术实现以及未来发展方向。

二、 项目概况

随着数字化时代的到来，图片已经成为人们生活中不可或缺的一部分。然而，许多用户在处理图片时受制于传统的软件工具，往往需要专业技能和复杂操作。为了解决这一问题，我们开发了一个网页 APP，旨在为用户提供一种简单易用的图片处理工具，使他们能够自由地编辑和分享图片。

本项目是一个基于 Flask 后端和 React 前端的数字图像处理应用。其主要目的是提供一个用户友好的平台，用户可以上传图像并对其进行各种处理。本项目后端使用 Flask 框架开发。Flask 是一个轻量级的 Python 框架，适合构建简单而强大的 Web 应用。在本项目中，Flask 负责处理用户请求、进行图像处理操作，并将结果返回给前端。在前端，我们使用了 React 框架。React 是一个流行的 JavaScript 库，专注于构建用户界面。在本项目中，React 负责构建动态和响应式的用户界面，使用户能够轻松上传图像、选择处理选项并查看结果。

三、 具体功能介绍

1、旋转

图 1: 原图

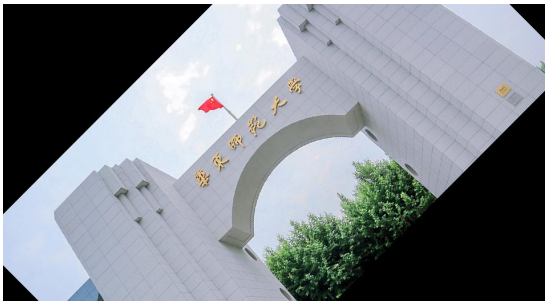

图 2: 旋转 45°

2、反转

图 3: 原图

图 4: 反转

3、缩放

3.1、放大

图 5: 原图，950x519px

图 6: 缩放至两倍，1900x1038px

3.2、缩小

图 7: 原图，950x519px

图 8: 缩放至 1/2，475x259px

4、加滤镜

4.1、模糊

图 9: 原图

图 10: 模糊

4.2、浮雕

图 11: 原图

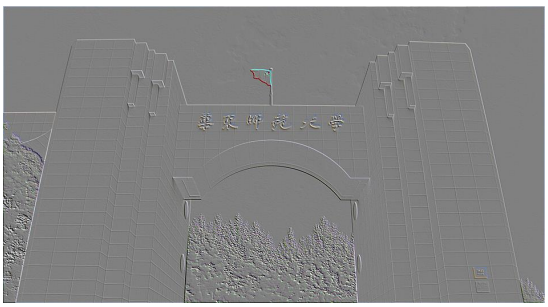

图 12: 浮雕

4.3、轮廓

图 13: 原图

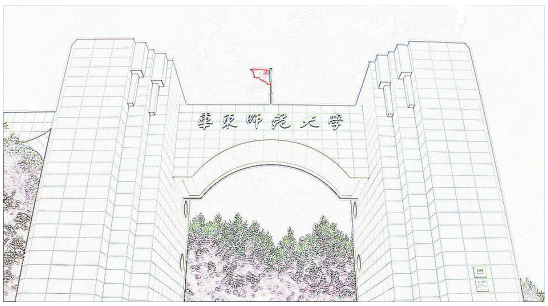

图 14: 轮廓

4.4、锐化

图 15: 原图

图 16: 锐化

5、调整颜色

图 17: 原图

图 18: 调整颜色，r2b2g0

6、对比度增强

图 19: 原图

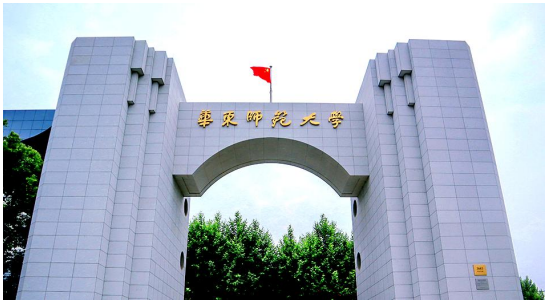

图 20: 对比度增强

7、平滑处理

图 21: 原图

图 22: 平滑处理

8、图像动作驱动

上传一张图片和一段视频，视频中的动作将会被应用到所选的图片上。囿于文档格式，这里只能以静态方式展示结果，实际上得到的结果为视频。下列图片中最左边为要应用的图片，中间为包含了指定动作的视频，右边为处理结果。

四、 部分关键算法展示

1、基本功能

本 APP 以 PIL 库为基础，提供了丰富多样的图片编辑功能，包括但不限于图片旋转、翻转、缩放的基础功能，也有为图片添加滤镜，调整图片色调，调节图像对比度以及图像平滑的进阶功能。用户可以根据自己的需求选择合适的编辑功能，对图片进行个性化处理。

```
if operation == 'rotate':
    image = image.rotate(float(parameter))
elif operation == 'flip':
    image = ImageOps.flip(image)
elif operation == 'scale':
    scale_factor = float(parameter)
    image = image.resize((int(image.width * scale_factor), int(image.height * scale_factor)))
```

向 python 中导入 PIL 库中的 Image 模块后，我们可以通过使用对应的函数来实现相应功能，用于旋转图像的 rotate 函数，翻转图像的 flip 函数，调整图像大小的 resize 函数。

```
elif operation == 'filter':
    if parameter == 'BLUR':
        image = image.filter(ImageFilter.BLUR)
    elif parameter == 'EMBOSS':
        image = image.filter(ImageFilter.EMBOSS)
    elif parameter == 'CONTOUR':
        image = image.filter(ImageFilter.CONTOUR)
    elif parameter == 'SHARPEN':
        image = image.filter(ImageFilter.SHARPEN)
```

如果选择了滤镜 (filter) 功能，则可以根据选择参数在模糊、浮雕、轮廓、锐化滤波中选择一项对图片进行处理。

```
elif operation == 'color_adjust':
    r, g, b = map(float, parameter.split(','))
    r = r if r else 1.0
    g = g if g else 1.0
    b = b if b else 1.0
    r_channel, g_channel, b_channel = image.split()
    r_channel = r_channel.point(lambda i: i * r)
    g_channel = g_channel.point(lambda i: i * g)
    b_channel = b_channel.point(lambda i: i * b)
    image = Image.merge('RGB', (r_channel, g_channel, b_channel))
```

通过向图像添加透明度的方法来实现对图像的色调调整。

```
elif operation == 'contrast':
    enhancer = ImageEnhance.Contrast(image)
    image = enhancer.enhance(2)
elif operation == 'smooth':
    image = image.filter(ImageFilter.SMOOTH)
```

除此以外，还能调节图像对比度以及用平滑滤波处理图像。

2、图像动作驱动

作为本 APP 的独特功能，用户可以使用本功能“让图片动起来”功能为静态图片增加动态效果。通过简单的设置和调整，用户可以使图片实现动画效果，增强视觉吸引力。

为实现图片动态化功能，我们引入了 First Order Motion Model(FOMM)[2][4] 这一模型。传统意义上的换脸是视频图像帧级别的操作，需要使用大量的双方人脸图像的数据进行事先训练，而 FOMM 可以在给定一个源人物、一个驱动视频的情况下生成一个新的视频。我们尝试了解了 FOMM 模型背后的原理，并试着作出大致归纳如下：

整个模型训练过程是图像重建的过程，输入是源图像和驱动图像，输出是保留源图像物体信息的含有驱动图像姿态的新图像，其中输入的两张图像来源于同一个视频，即同一个物体信息，那么整个训练过程就是驱动图像的重建过程。

2.1、FOMM 的基本原理

大体上来说 FOMM 分成两个模块，一个是运动估计模块，另一个是图像生成模块。其中运动估计模块可以进一步细分成粗略运动估计和精细运动预测。粗略运动被建模为独立物体部件之间的稀疏运动，而密集运动则生成整个图像的光流和置信度图。这里分别用 $\mathbf{S}$ 和 $\mathbf{D}$ 表示从同一个视频中提取的源帧和驱动帧。

第一步是从 $\mathbf{S}$ 和 $\mathbf{D}$ 中估计粗运动。每个物体部分的运动由仿射变换 $A^k_{\mathbf{X}\leftarrow\mathbf{R}} \in \mathcal{R}^{2\times 3}$ 表示。这些变换都是相对于一个抽象的公共参考框架 $\mathbf{R}$，$\mathbf{X}$ 代表 $\mathbf{S}$ 或者 $\mathbf{D}$。

运动是为 k 个不同的部件进行估计的。一个编码器-解码器关键点预测网络对输入图像输出 k 个热图 $\mathbf{M}^1, \cdots, \mathbf{M}^k$ 然后进行 softmax 处理，使得 $\mathbf{M}^k \in [0,1]^{H\times W}$，其中 H 和 W 分别是图像的高度和宽度，且对于所有像素位置 z（图像中的 (x, y) 坐标），存在 $\sum_{z\in\mathcal{Z}} \mathbf{M}^k(z) = 1$ 其中 $\mathcal{Z}$ 是所有像素位置的集合，$\mathbf{M}^k(z)$ 是像素 z 上第 k 个热图的权重。因此，仿射变换的平移分量（即 $A^k_{\mathbf{X}\leftarrow\mathbf{R}}$ 的最后一列）可以通过 softargmax 进行估计：

$$\mu^k = \sum_{z\in\mathcal{Z}} \mathbf{M}^k(z) z \tag{1}$$

在 FOMM 中，其余的仿射参数是逐像素回归的，并形成 4 个额外的通道 $P^k_{ij} \in \mathcal{R}^{H\times W}$，其中 $i \in \{0, 1\}, j \in \{0, 1\}$ 是仿射矩阵 $A^k_{\mathbf{X}\leftarrow\mathbf{R}}$ 的索引。通过加权池化来估计这些参数：

$$A^k_{\mathbf{X}\leftarrow\mathbf{R}}[i, j] = \sum_{z\in\mathcal{Z}} \mathbf{M}^k(z) P^k_{ij}(z) \tag{2}$$

FOMM 将这种计算运动的方式称为基于回归的方法，其中仿射参数由网络预测并进行池化以计算 $A^k_{\mathbf{X}\leftarrow\mathbf{R}}$。然后通过公共参考框架计算 $\mathbf{D}$ 和 $\mathbf{S}$ 之间的部件 k 的运动：

$$A^k_{\mathbf{S}\leftarrow\mathbf{D}} = A^k_{\mathbf{S}\leftarrow\mathbf{R}} \begin{bmatrix} A^k_{\mathbf{D}\leftarrow\mathbf{R}} & \\ 0 \quad 0 & 1 \end{bmatrix}^{-1} \tag{3}$$

给定粗略的运动，下一步是预测光流和置信度图。由于已知每个部件之间 $\mathbf{D}$ 和 $\mathbf{S}$ 的变换，任务是将这些变换结合起来以获得一个单一的光流场。为此，光流预测器为每个像素选择适当的粗略变换，通过对粗略运动的加权求和来完成。形式上，输出 $K + 1$ 个分配图，每个区域一个，加上背景，FOMM 假设背景是静止的，然后在这些分配图上逐像素地应用 softargmax，使得 $\mathbf{W}^k \in [0,1]^{H\times W}$，$W^0$ 对应于背景，且对于所有像素

$\sum_{k=0}^{K} \mathbf{W}^k(z) = 1, \forall z$ 每个像素的光流 $\mathbf{O}(z) \in \mathbb{R}^2$ 可以计算为：

$$\mathbf{O}(z) = \mathbf{W}^0(z)z + \sum_{k=1}^{K} \mathbf{W}^k(z) A^k_{\mathbf{S}\leftarrow\mathbf{D}} \begin{bmatrix} z \\ 1 \end{bmatrix} \tag{4}$$

在这样的模型中，当背景有轻微运动时，动画制作变得具有挑战性。FOMM 模型通过将几个可用的关键点分配给背景来自动适应这种情况。

同样的网络也会预测一个置信度图 $\mathbf{C} \in [0,1]^{H\times W}$ 以处理源图像中缺失的部分。最后，将 S 通过编码器，接着使用光流（方程 4）对生成的特征图进行变形，并乘以置信度图。解码器随后重建驱动图像 D。

2.2、基于主成分分析 (PCA[5]) 的动作估计

准确的运动估计是高质量图像动画的主要要求。如前所述，FOMM 回归了仿射参数。这需要更高容量的网络，并且泛化能力较差。FOMM 提出了一种不同的运动表示方法：所有的运动都是直接从热图 $\mathbf{M}^k$ 中测量的。FOMM 像之前一样计算平移分量，而在平面内的旋转和 $x-$ 和 $y-$ 方向上的缩放则通过对热图 $\mathbf{M}^k$ 进行主成分分析（PCA）来计算。形式上，第 k 个区域从参考框架到图像的变换 $A^k_{\mathbf{X}\leftarrow\mathbf{R}} \in \mathbb{R}^{2\times 3}$ 计算如下：

$$\mu^k = \sum_{z\in\mathcal{Z}} \mathbf{M}^k(z) z \tag{5}$$

$$U^k S^k V^k = \sum_{z\in\mathcal{Z}} \mathbf{M}^k(z)(z-\mu^k)(z-\mu^k)^{\mathrm{T}} (\mathrm{SVD}) \tag{6}$$

$$A^k_{\mathbf{X}\leftarrow\mathbf{R}} = \left[U^k S^{k\,\frac{1}{2}}, \mu^k\right] \tag{7}$$

这里使用了奇异值分解（SVD）方法来计算主成分分析（PCA），方程 (6) 将热图的协方差分解为单位矩阵 U^k 和 V^k，以及奇异值对角矩阵 S^k。我们称这种方法为基于 PCA 的方法，以区别于方程 (1) 中的基于回归的方法。尽管在这里使用了相同的区域表示和编码器，但由于论文中提出的前景运动表示方法（如上所述），编码后的区域在表示上存在显著差异，这里的方法可以将运动映射到有意义的物体部件上，例如具有关节的身体的四肢。值得注意的是，在我们的基于 PCA 的方法中，剪切变换没有被捕获，因此这里的变换不是完全的仿射变换，只有五个自由度而不是六个。然而，这种方法足以捕获任务所需的运动，其中剪切是仿射变换中的一个较不重要的组成部分。

在这两种方法中，参考框架都仅作为源图像和驱动图像坐标框架之间的中间坐标框架。然而，在这里（与 FOMM 相比），它实际上不是抽象的，而是对应于热图被白化的坐标框架（即均值为零，协方差为单位矩阵的坐标框架）。然后按照方程 (3) 计算 $A^k_{\mathbf{S}\leftarrow\mathbf{D}}$。

2.3、背景动作估计

背景占据了图像的大部分。因此，即使是帧之间的小背景运动，例如由于相机运动，也会对动画质量产生负面影响。FOMM 不将背景运动单独处理，因此必须使用关键点来对其建模。这有两个负面影响：（1）需要额外的网络容量，因为使用了多个关键点来建模背景而不是前景；（2）对训练集的过拟合，因为这些关键点专注于背景的特定部分，这些部分可能在测试集中不会出现。因此，FOMM 使用一个编码器网络来额外预测一个仿射背景变换 $\mathbf{A}^{\mathbf{0}}_{\mathbf{S}\leftarrow\mathbf{D}}$，假设 $\mathbf{S}$ 和 $\mathbf{D}$ 作为输入并预测六个实值 $a_1, \cdots, a_6$，使得 $\mathbf{A}^{\mathbf{0}}_{\mathbf{S}\leftarrow\mathbf{D}} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \end{bmatrix}$。由于 FOMM 的框架是无监督的，背景网络可以将前景的部

分内容包含到背景运动中。在实际应用中，这种情况不会发生，因为网络更容易使用基于 PCA 的运动表示来处理前景。网络也更简单地使用 **S** 和 **D** 来预测背景运动，而不是在热图中对前景进行建模。

2.4、图像生成

类似于 FOMM，改进过的模型将目标图像渲染为两个阶段：一个逐像素的光流生成器将粗略运动转换为密集的光流，然后根据光流对源图像的特征进行变形，最后进行缺失区域的修复。密集光流预测器的输入是一个 $H \times W \times (4K + 3)$ 的张量，其中每个区域有四个通道，三个通道用于根据区域的仿射变换对源图像进行变形，一个通道用于区域的热图，热图是 $\mathbf{M}^k$ 的高斯近似，还有三个通道用于根据背景的仿射变换对源图像进行变形。与 FOMM 不同，这里从热图中估计协方差，而不是使用常数方差。密集光流由以下公式给出（参见方程 (4)）：

$$\mathbf{O}(z) = \mathbf{W}^0(z)z + \sum_{k=1}^{K} \mathbf{W}^k(z) A^k_{\mathbf{S}\leftarrow\mathbf{D}} \begin{bmatrix} z \\ 1 \end{bmatrix} \tag{8}$$

预测的光流和置信度图像按照 FOMM 进行使用。然而，与 FOMM 不同，但类似于 Monkey-Net[1] ，在这里使用了可变形跳跃连接连接 [3] 编码器和解码器。

五、 总结与未来展望

本项目的成功实施展示了一个功能全面且用户友好的数字图像处理网页应用。我们通过应用 Flask 和 React 框架，开发了一个具备多种图像编辑功能的网页 APP，涵盖了从基本的旋转、翻转、缩放，到进阶的滤镜效果、颜色调整、对比度增强以及图像平滑处理等功能。同时，我们还实现了独特的“图像动作驱动”功能，使静态图像能够通过驱动视频获得动态效果，极大地拓展了图像处理的应用场景。

在技术实现方面，我们基于 PIL 库提供了多种图像处理功能，通过整合 FOMM 模型实现了图像动画效果，展示了先进的计算机视觉技术在实际应用中的效果。这些技术的应用不仅提高了用户体验，也为今后的开发提供了丰富的实践经验。

未来，我们计划对本项目进行进一步的优化和扩展。首先，我们将致力于提升“图像动作驱动”功能的效果，探索更多的模型和技术以增强图像动画的质量和灵活性。其次，我们计划增加更多的图像编辑功能，例如智能抠图、背景移除等，以满足用户更多样化的需求。此外，我们还将关注用户反馈，持续改进用户界面和交互体验，以使应用更加便捷和高效。

在技术层面，我们将进一步探索深度学习技术在图像处理中的新应用，例如引入生成对抗网络（GANs）进行图像增强和修复，或者使用迁移学习方法提升模型的泛化能力。随着图像处理技术的不断进步，我们相信本项目将能够不断创新和发展，为用户提供更多优质的服务和功能。

总的来说，本项目不仅展示了我们在数字图像处理方面的学习成果，也为未来的进一步学习奠定了坚实的基础。我们期待在未来的研究学习中，能够继续紧随技术创新，开发出功能更为完备的系统。

参考文献

[1] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. Animating arbitrary objects via deep motion transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[2] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. *arXiv preprint*, 2020.

[3] Aliaksandr Siarohin, Enver Sangineto, Stéphane Lathuilière, and Nicu Sebe. Deformable gans for pose-based human image generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[4] Aliaksandr Siarohin, Oliver J. Woodford, Jian Ren, Menglei Chai, and Sergey Tulyakov. Motion representations for articulated animation. *arXiv preprint*, 2021.

[5] Michael E. Wall, Andreas Rechtsteiner, and Luis M. Rocha. Singular value decomposition and principal component analysis. In *A Practical Approach to Microarray Data Analysis*, pages 91–109. Springer, 2003.