CV 2022 Fall

Project: A look into the Past

Jiacheng Xu PB20061368, Zeyu Zhang PB19061215

Completion Date ..... Dec 1, 2022
Mentor ..... Yang Cao

Abstract

在本次 CV 课程的项目中，我们进行的是第一个任务：*A Look into the past*，主要实现了将一张图像（一般是距离现在有一定时间的老图片，也多为黑白色，称为 past）中选择出合适的部分，拼接到另一张图像（一般是现在拍摄的彩色图片，称为 present）上。在使用了单应性变换、*SIFT* 特征点匹配、显著图等技术后，我们实现了图片之间视角的转换、区域的匹配等基础功能，同时处理了许多细节上的问题，如边缘处理。在此基础上，我们进一步将图片与图片之间的匹配扩展至图片与视频的匹配，具体包括一张图片和一个视频的匹配，以及两段视频之间的匹配，均取得了不错的效果。

1 Introduction

在面对老图片，尤其是一些历史感浓厚的老照片的时候，人们常常会想：如果过去的场景出现在现在会是什么样的。针对这个需求。我们尝试将一些过去拍摄的“老图片”拼接到现在拍摄的同一场景的“新照片”上。这个任务看似简单，但是存在许多细节问题。首先是照片拍摄视角不同，即使是同一场景，但是由于视角的不同，如果单纯的拼接会导致图像变形，边缘错位等一系列问题。我们通过单应性变换解决这个问题。其次是选取合适的场景进行拼接，我们希望将 past 中的关键部分，如人、标志性建筑等，拼接到 present 上，但是如果选取区域不合适，则会失去期望的效果，单纯地变成两张图片的叠加。我们通过 *SIFT* 特征点匹配、显著图、以及目标检测（*YOLO* 等多种方法来选择合适的区域进行自适应区域选择。最后，我们将这一技术扩展到视频，将 past 匹配到一段 present 视频中，实现了类似“历史照进现实”的效果。

2 Formulation and Theory

A Look into the past 任务可以建模成图像拼接问题，其中主要涉及如下技术。

2.1 SIFT 特征点匹配

尺度不变特征转换 *(Scale Invariant Feature Transform, SIFT)* 是图像处理领域中的一种局部特征描述算法，其尺度不变性非常适合图像之间的匹配。此外，当旋转图像，改变图像亮度，移动拍摄位置时，仍可得到较好的效果。由于篇幅原因，仅介绍算法大致流程及要点。

1. 首先需要为图像创建图像金字塔，以表示不同尺度下图像的特征，越底层的图像，保留图像的细节信息越多，反之，越靠近顶层，图像的整体特征更为突出。然后需要确定图像中的极值点位置。

2. 通过极值求解关键点。为了反应多尺度的特点，我们对不同尺度的子图像进行差分处理，获得高斯差分金字塔，然后在其中寻找极值点。注意，除了考虑 x, y 方向的点，还要考虑尺度方向的点，所以判断一个像素点是否为极值点，要与周围的 26 个点（$9 \times 2 + 8$）进行比较。

3. 确定关键点方向。统计以特征点为圆心，以该特征点所在的高斯图像的尺度的 1.5 倍为半径的圆内的所有的像素的梯度方向及其梯度幅值，并做 1.5σ 的高斯滤波(高斯加权，离圆心也就是关键点近的幅值所占权重较高)。

4. 生成关键点描述符。对于每个关键点，首先划定该点的区域（一个圆形区域，半径一般限于周围的 4 个子块），然后将区域划分为 4×4 的子块，对每一子块进行 8 个方向的直方图统计操作，获得每个方向的梯度幅值，总共可以组成 128 维描述向量。

2.2 单应性变换

单应性变换（*Homograph*）主要是为了解决图片拼接时的视角问题。在此首先给出单应性的定义：

Definition 2.1. 用无镜头畸变的相机从不同位置拍摄同一平面物体的图像之间存在单应性，可以用投影变换表示。

有了定理作为保证，我们就可以通过计算单应性矩阵来求得两张图片之间的投影变换关系，具体如公式1所示：

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H_{3\times3} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{1}$$

其中，

$$H_{3\times3} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tag{2}$$

为了求解该单应性矩阵，我们需要寻找对应的特征点进行匹配构造单应性方程。根据单应性矩阵的自由度为 8，理论上我们仅需要 $8/2 = 4$ 组对应点就可以求解出此矩阵。但是为了减少匹配的误差，我们选取多组特征点，先进行 *KNN* 聚类处理，一张图中每个特征点选择出另一张图中匹配指标最大的对应点进行匹配。同时使用 RANSAC 算法对匹配进行约束，剔除错误匹配。对于一个特征点，如果第一匹配和第二匹配的点之间的指标相差不大（实验中给定的指标为 $l_2 <= 0.7l_1$，其中 l_i 为匹配指标），说明该特征点特征还不够显著，故放弃该匹配。

2.3 泊松编辑

泊松编辑的主要思想是在以原图像块内梯度场为指导，将融合边界上目标场景和源图像场景的差异平滑地扩散到融合图像块中，这样的话融合后的图像块能够无缝地融合到目标场景中，并且色调可以与目标场景相一致。公式化的描述如下：

$$I^* = \arg min_f \iint_\Omega \|\nabla I - \nabla I_A\|^2, \text{ s.t. } I^* \mid_{\partial\Omega} = I_B \mid_{\partial\Omega} \tag{3}$$

于是问题转化为求解一个 Dirichlet 边界条件下的泊松方程。根据欧拉公式，有

$$F_f - \frac{\partial}{\partial x} F_{f_x} - \frac{\partial}{\partial y} F_{f_y} = 0 \tag{4}$$

将公式4代入公式3，得到

$$\Delta I = \text{div}(\nabla I_A), \ s.t. \ I \mid_{\partial\Omega} = I_B \mid_{\partial\Omega} \tag{5}$$

从公式中可以看出，无缝融合的实现关键在于边界的连续性，目标函数中梯度差最小化正是反映了这一点。

3 Model and Code

3.1 如何匹配两张图片

我们选择 SIFT(Scale-invariant feature transform) 算法来提取兴趣点，因为它具有尺度不变性，所以可以很容易地拓展到多尺度问题上。其次我们使用了 KNN 算法进行聚类，挑选其中较好特征点对。

```
surf = cv2.SIFT_create()
# find the keypoints and descriptors with SIFT
kp1, des1 = surf.detectAndCompute(past, None)
kp2, des2 = surf.detectAndCompute(present, None)

# BFMatcher with default params
bf = cv2.BFMatcher()
matches = bf.knnMatch(des1, des2, k=2)

# Apply ratio test, less is better
good = []
for m, n in matches:
    if m.distance < 0.7 * n.distance:
        good.append(m)
```

3.2 如何进行视角的变换

如果在第一步中，我们得到了足够多的质量好的特征点对，就可以求解对应的单应性方程，得到单应性矩阵，从而进行单应性变换。在实施中，我们使用了 RANSAC 算法来挑选“好”的特征点。

```
    src_pts = np.float32([kp1[m.queryIdx].pt for m in good]).reshape(-1,1,2)
    dst_pts = np.float32([kp2[m.trainIdx].pt for m in good]).reshape(-1,1,2)
    H, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC, 5.0)
```

3.3 如何选取合并的区域

为了选取 past 中合适的部分进行拼接，我们提出了三种可行的方法：前一种是人工方法，可拓展性不强；后两种是自适应方法，可拓展性强。

3.3.1 人为选取

实现方法：基于单应性矩阵我们可以得到一些特征点，选取特征点的中心作为圆心，在 past 上绘制一个椭圆，融合到 present 上 (在融合中，我们还使用了引导滤波来模糊边界)。之所以选取椭圆，是因为我们希望最后的效果能够有电子相册的质感。但是这种方法可拓展性太弱，所以紧接着我们又提出另外两种方法。

```
    src_center = np.mean(src_pts, axis=0, dtype=np.int32)
    radius = (src.shape[0] // 3 , src.shape[1] // 3)
    dst_center = np.dot(H, np.array([src_center[0][0], src_center[0][1], 1]))
    dst_center = dst_center / dst_center[2]
    dst_center = dst_center.astype(np.int32)[:2]
    area = np.ones_like(dst)
    area = cv2.ellipse(area, tuple(dst_center), tuple(map(lambda x : int(0.9*x), radius
    )), angle=90, startAngle=0, endAngle=360, color=0, thickness=-1)
    dst = area * dst
    blur = np.copy(result)
    blur[area==0] = 0
    crop = np.copy(result)
    crop[area==1] = 0
    result = cv2.ximgproc.guidedFilter(dst, blur, 33, 2, -1) + crop
```

3.3.2 显著图方法

Saliency map 可以把图像简单化，并且经过一定的处理，可以生成存在潜在目标的区域，最大的优点是速度较快。我们使用 SpectralResidual 算法计算显著图，根据阈值生成 object map 在这之后使用泊松编辑来完成融合 (关于泊松融合，具体见3.4小节。)。但是，对于噪声较大的图像，saliency map 的表示能力有限，而且检测出的目标往往是分散且破碎的，不符合我们的任务需求。于是我们提出了第三种方法：目标检测。

```
saliencyAlgorithm = cv2.saliency.StaticSaliencySpectralResidual_create()
(success, saliencyMap) = saliencyAlgorithm.computeSaliency(img)
biSaliencyMap = (saliencyMap > threshold * maximum).astype(np.uint8)
mask  = masked_biSaliencyMap * 255
result = cv2.seamlessClone(past, present, mask, (past.shape[1]//2, past.shape
 [0]//2), cv2.NORMAL_CLONE)
```

3.3.3 目标检测

对于很多场景而言，中心都在人等物体身上，我们也愿意把人等物体迁移到不同图像上。我们使用在 coco 数据集上训练了 100 个 epoch 的 YOLOv4-tiny 模型来提取 bounding box，随后用一个 box 包括所有的物体，作为候选区域。最后，我们使用泊松编辑进行融合。(关于泊松融合，具体见3.4小节。)

```
r_image, boxes = yolo.detect_image(image, crop, count)
pos_list = []
for box in boxes:
    pos_list.append(convert(box))  # convert() function receive a box(tensor) and
 return four points(array) top, left, bottom, right
top = min(pos_list[i][0] for i in range(len(pos_list)))
left = min(pos_list[i][1] for i in range(len(pos_list)))
bottom = max(pos_list[i][2] for i in range(len(pos_list)))
right = max(pos_list[i][3] for i in range(len(pos_list)))
box = [top, left, bottom, right]

mask = np.ones((box[2]-box[0], box[3]-box[1], 3), np.uint8) * 255
center = ((box[1] + box[3]) // 2, (box[0] + box[2]) // 2)
result = cv2.seamlessClone(src, dst, mask, center, cv2.NORMAL_CLONE)
```

3.4 如何模糊边界

我们尝试过多种滤波方式，包括高斯滤波、中值滤波、双边滤波等，最后发现在一定条件下 (先做滤波再进行融合) 引导滤波效果较好；同时我们也使用了泊松编辑来进行融合，效果也很喜人。

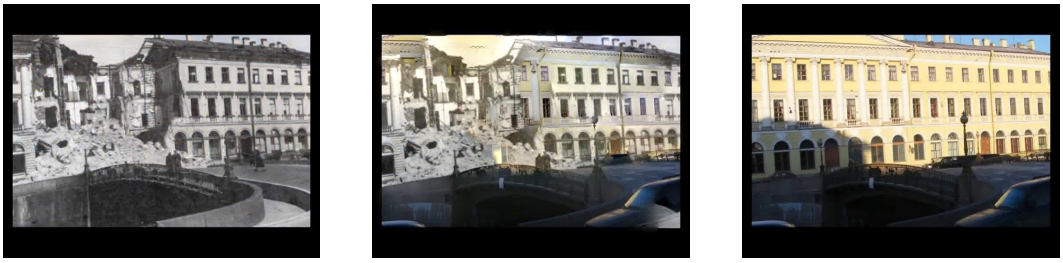
4 Results and Analysis

图1是一张基于 SIFT 算法和人为选区合并的示例图片。可以看到，经过单应性变换后，past 可以被比较好地拼接到 present 对应的位置。此外，由于经过了一次引导滤波，拼接区域的黑边也几乎不见了。

(a) past (b) result (c) present

Figure 1: manual region selection

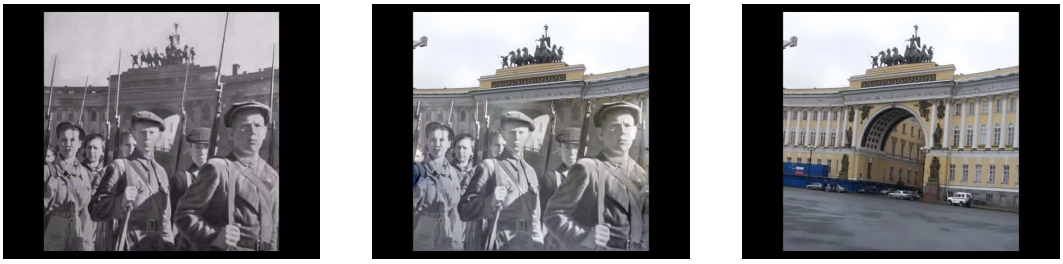
图2是一张基于 saliency map 的合成图。用 saliency map 提取的区域支离破碎，不规则性、不连续性很强，我们曾经尝试使用 canny 算法，基于边缘检测的方式拓展、融合这些区域，效果并不是很好，并且违背了我们使用 saliency map 的初衷：尽可能快地提取到 past 中可能存在物体的区域，将其合并到 present 中。但是我们也承认这是我们目前工作中的不足之处，在后续可以进行改进。



(a) past (b) result (c) present

Figure 2: saliency map region selection

图3基于 object detection 的合成图。我们使用在 coco 数据集上训练的预训练 YOLOv4 模型来完成目标检测的任务，然后基于从 past 图像提取的 bounding box 确定候选区，最终采用泊松编辑的方法合并到 present 图像。在这个问题中，我们曾经尝试过进行滤波进行去边操作，但是结果不尽如人意，这与我们的 past 和 present 图像时间跨度大，背景变化较大有关。



(a) past (b) result (c) present

Figure 3: object detection region selection

5 Conclusion

在这项工作中，我们结合了多种算法，如 SIFT, RANSAC, KNN, Guided Filter, SpectralResidual, YOLOv4 model, Possion Editing 等，以达到灵活的、不错的图像融合的效果。并且我们还提出了一个富有新意的想法 looks into the past，即用一张 pa 图像匹配一段 pr 视频，或者一段 pa 视频匹配一张 pr 图像，实现历史中的人物走在当下场景的效果。

但是这项工作同时也面临一些不足：例如对于 pa 图像中的候选区，与 pr 图像融合时，我们选择了 Normal Clone，如果 pr 图像中该区域已经有物体，甚至和 pa 图像中的物体是同一个物体，显然 Mixed Clone 是更好的做法，所以更好的做法是先检测待融合的区域，再根据检测结果和我们的需要挑选融合方式；使用 saliency map 提取区域时，提取的区域过于无序，因此后续可以进行融合、消弭等改进操作；对于我们提到的 looks into the past，我们目前仅处于浅尝辄止的阶段，还有很多实施时的具体细节做得不够好，改进空间很大。

6 Acknowledgement

通过一学期的学习，我们收获了许多，对于计算机视觉的重点和前沿技术有了更加深刻的了解，同时处理数字图像的能力也得到了较为显著的增强。感谢老师和助教一学期以来的辛勤付出！