浙江大學

计算机科学
与技术学院

COLLEGE OF COMPUTER SCIENCE AND TECHNOLOGY OF ZHEJIANG UNIVERSITY

基于姿态的个性化可控的武打动作生成方法结项答辩

答辩人姓名

我们的研究团队

Our Research Team

务求实学，存是去非

吴家驹 / 项目组长

计算机科学与技术

黄可欣 / 项目成员

软件工程

尹浩柏 / 项目成员

软件工程

正其谊、不谋其利，明其道、不计其功

目录

CONTENTS

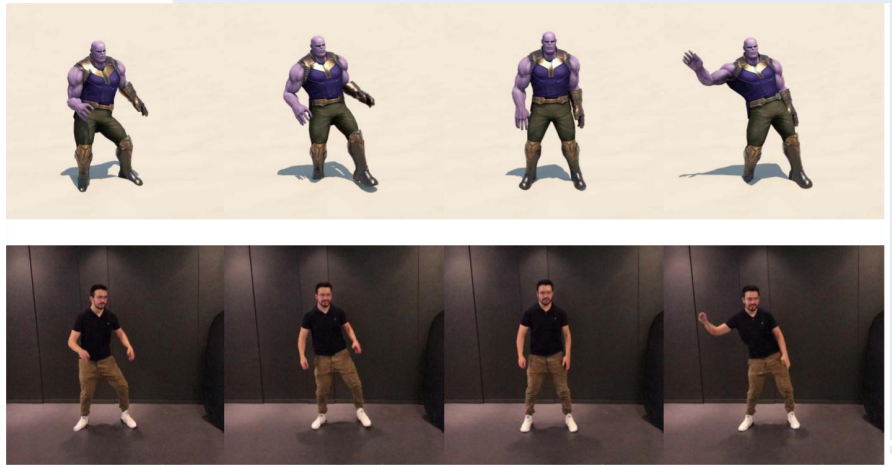
01

项目背景及简介

答辩人：吴家驹

01 选题背景

视频行业如今发展越来越快，视频行业的市场和用户数目不断扩张，这也使得视频特效的需求随之增长。尤其是有武打动作的特效视频，能带给用户很好的体验感，在电影、电视剧、自媒体视频、以及游戏动画等方方面面都有很大的需求



为了能更便捷有效地生成武打视频，我们的项目便致力于设计一套流程，能把武打姿态整合为视频集，根据输入的人物外观，自动生成武打动作视频，实现基于姿态的个性化可控的武打动作生成。 如果实现了基于姿态的个性化可控的武打动作生成，那么武打动作的制作就可以成为量产化的工作。

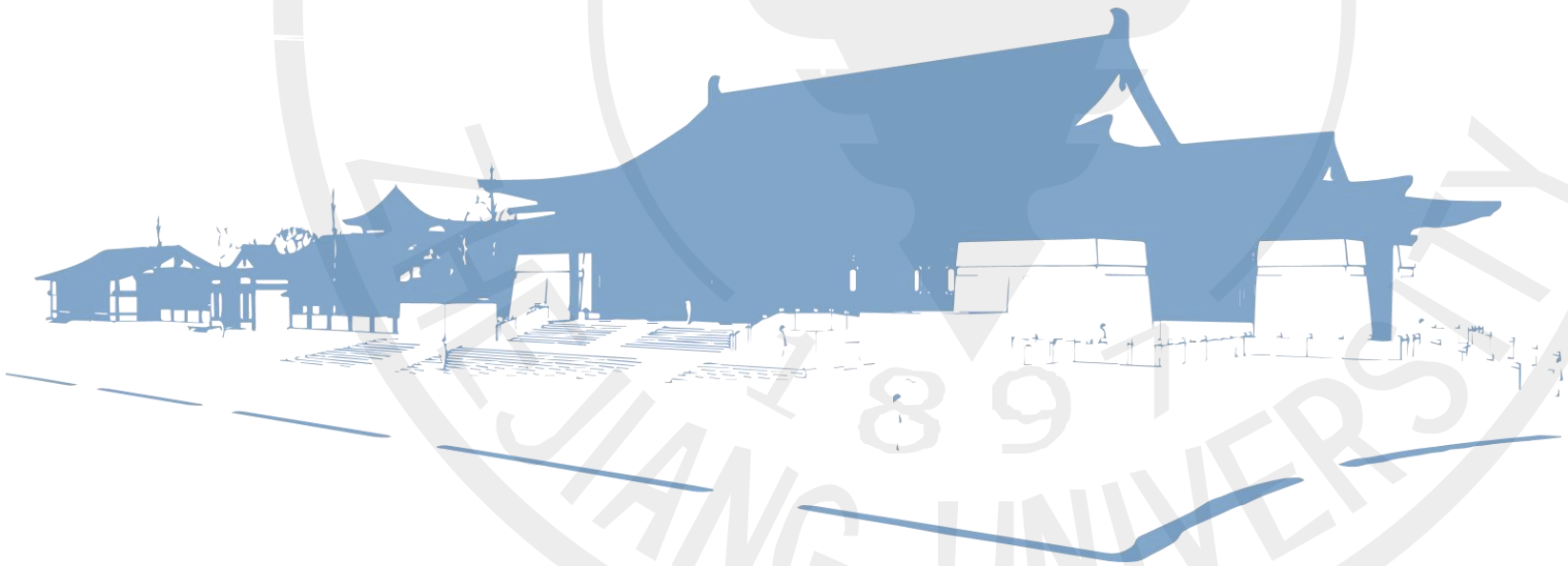
01 项目简介

项目目标：构建出一套能实现基于姿态的个性化可控的武打动作生成的程序，能有效地实现生成个性化武打动作视频的功能。

项目内容：首先完成了一套姿态可控的武打动作迁移的流程，主要任务是把武打动作从源视频中迁移到目标视频的人物中，并且要实现姿态的可控。此外，该项目还建立了一个简单的武打动作库，存储了一些已有的武打动作骨架。在原有项目的基础上增加了关节节点的修改功能，完成了根据两帧动作，完成中间30帧平滑过渡的自动生成。这使得生成新的武打动作不需要对每一帧的关节信息进行创建，只需要生成关键帧的关节信息，就可以平滑生成整个武打动作的骨架。

02

项目过程与成果展示



02 项目发展过程

务求实学，存是去非

正其谊、不谋其利，明其道、不计其功

2022/09

01 项目学习阶段

实现具有数字识别功能的经典卷积神经网络LeNet-5

2022/11

02 项目中期答辩

2023/02

03 动作迁移模块

我们使用小组成员的图片和准备好的太极动作视频进行输入，得到了小组成员完成太极动作的视频

2023/03

04 构建动作库 修改骨架关节

存储了一些已有的武打动作骨架，在原有项目的基础上增加了关节节点的修改功能。

2020/04

05 新的武打动作的可控化生成

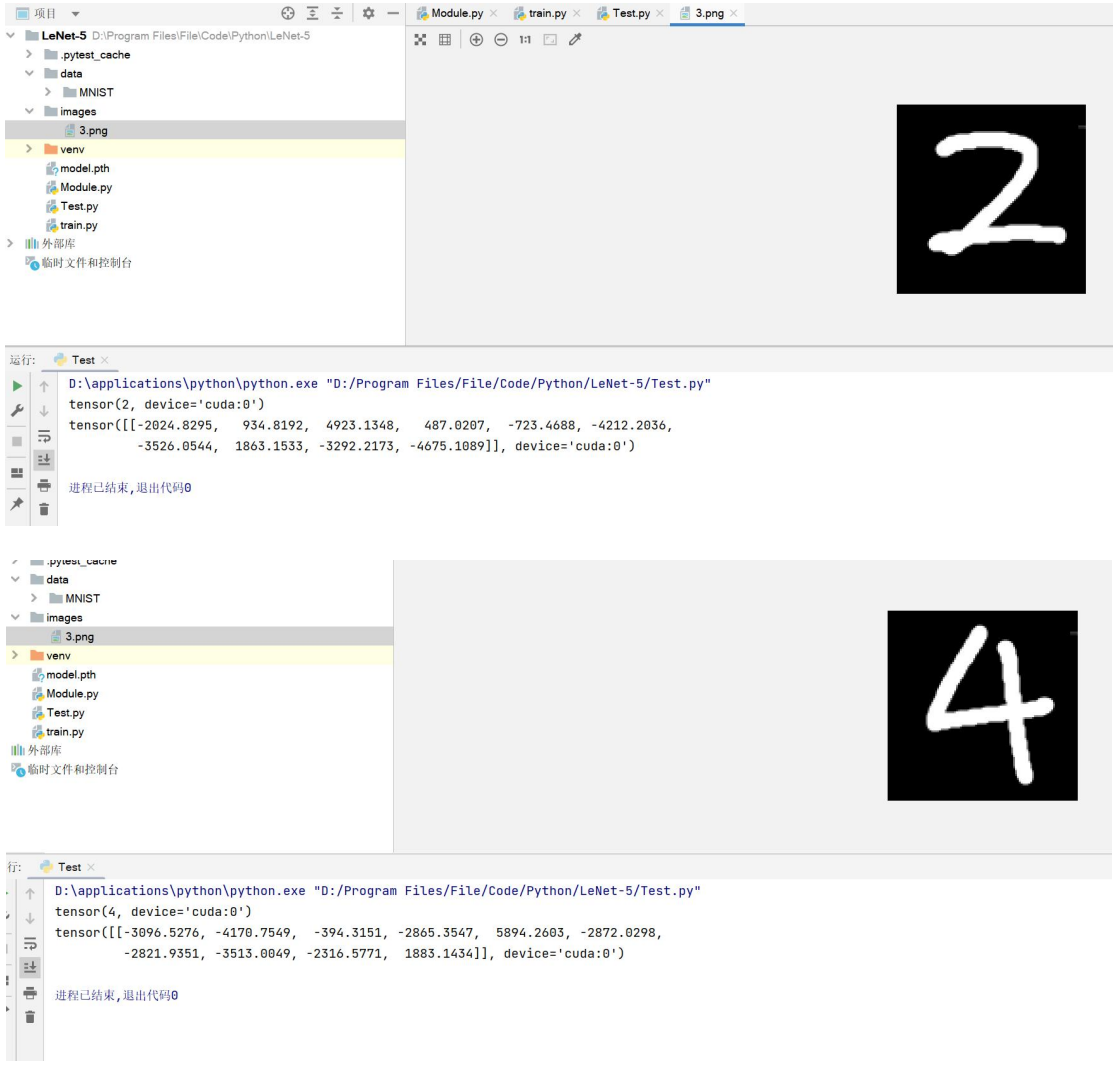
完成了根据两帧动作，完成中间30帧平滑过渡的自动生成。

02 成果展示

务求实学，存是去非

项目学习阶段

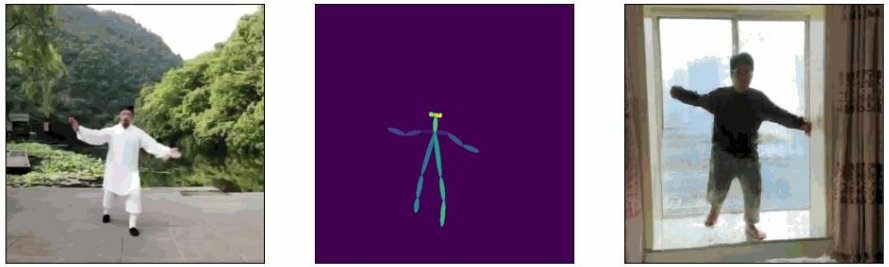
在项目进行的初期阶段，观看过详细教程后，经过小组三位成员讨论沟通决定本阶段的主要任务目标是经典卷积神经网络LeNet-5的实现第二阶段，实现具有数字识别功能的LeNet-5卷积神经网络。



正其谊、不谋其利，明其道、不计其功

动作迁移模块

该模块的基本流程是，首先从源视频中进行姿态估计，提取动作骨架，生成骨架团以及提取出关节的位置信息，第二步是提取目标视频的骨架信息，经过整理图片大小之后，训练一个针对目标视频的动作生成模型。然后利用训练好的模型和待迁移的武打动作的骨架图片，进行动作迁移。最后把动作迁移之后生成的帧整理成GIF格式图片，然后转成视频格式并配上音乐。

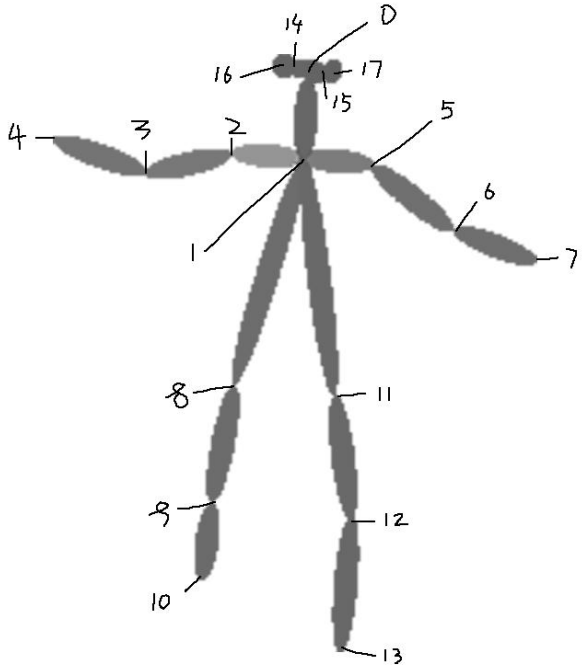


动作库的构建 关节修改

我们建立了一个简单的武打动作库，存储了一些已有的武打动作骨架，我们可以根据已经存储的动作库骨架，为目标快速生成武打动作视频。

动作库的构建 关节修改

我们在原有项目的基础上增加了关节节点的修改功能，我们增加了关节坐标的二维张量的加载与保存，利用npy文件存储，可以利用python文件对每一帧对应的关节位置张量进行修改，完成动作数据的可控修改。

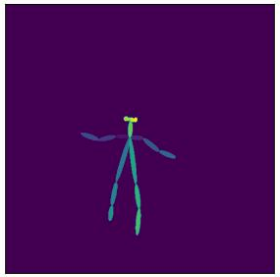


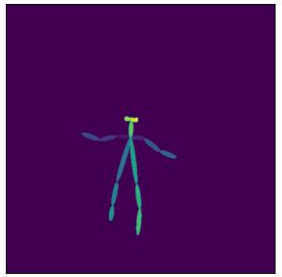
新的武打动作的可控化生成

我们完成了根据两帧动作，完成中间30帧平滑过渡的自动生成，以便于我们可以创造新的动作。这使得生成新的武打动作，不需要对每一帧的关节信息进行创建，只需要生成关键帧的关节信息，就可以平滑生成整个武打动作的骨架。

开始帧

结束帧



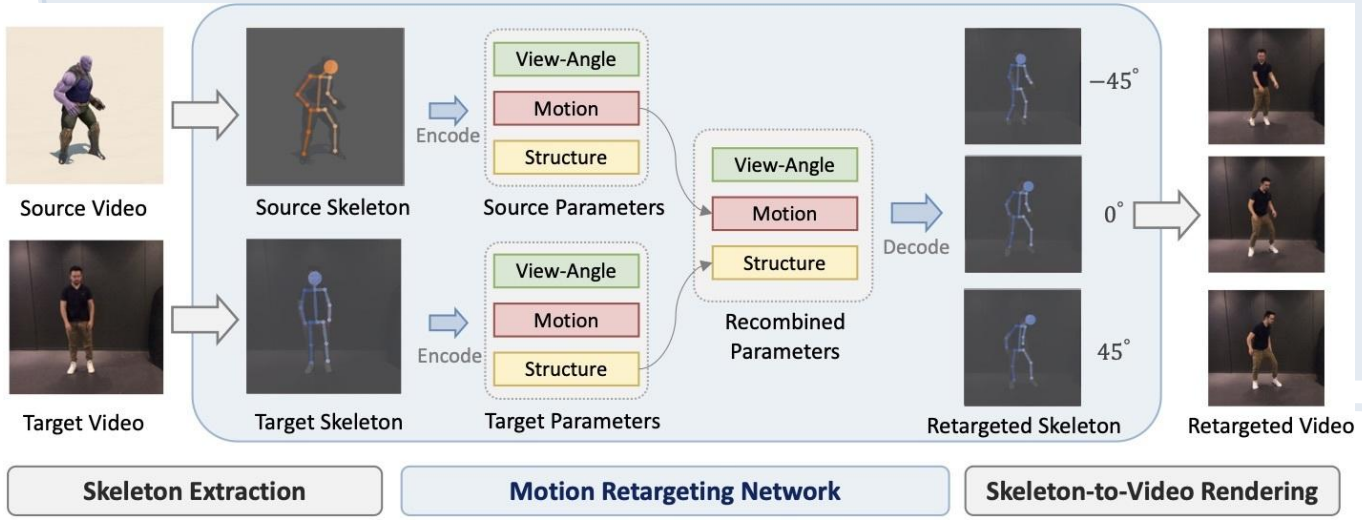


03

项目实现方法

答辩人：吴家驹

Transmom的整体流程大概为：提取source和target人体骨架（关节）序列，通过MRN得到source的Motion特征和target的Structure特征组合在一起，然后模型输出一个3D的骨架（关节）序列，再根据不同的视角投射到2D骨架图序列上，最后通过一个预训练的img2img网络将2D骨架序列转变到图像域中得到最后的输出。



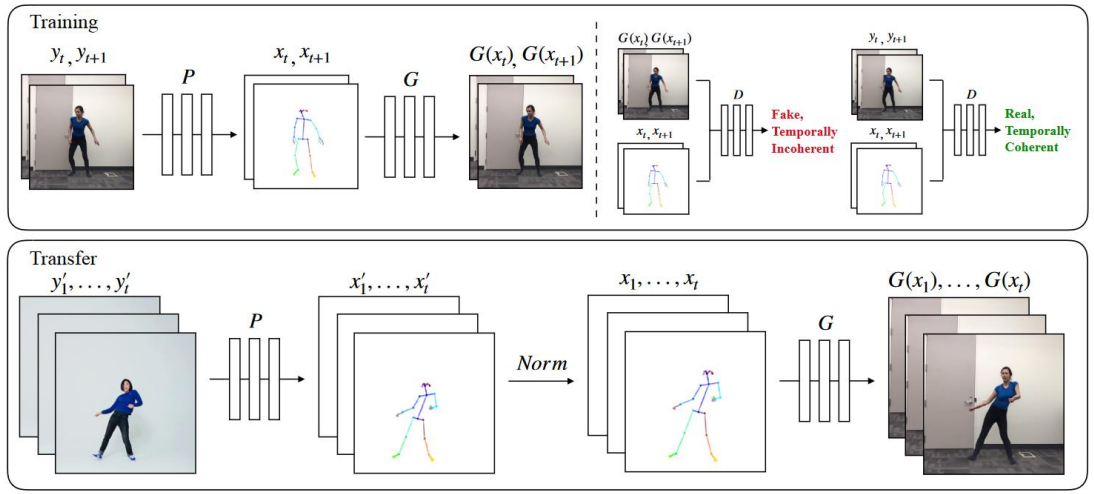
03 Everybode Dance Now

动作迁移管道分为三个部分：

姿态检测：在姿态检测阶段，通过预训练的姿态检测模型（openpose）从源视频中描绘给定帧的姿态图形。

全局姿态标准化：全局姿态标准化阶段，计算给定帧内源和目标人物身体形状和位置之间的差异，将源姿态图形转换到符合目标人物身体形状和位置的姿态图形。

从标准化后的姿态图形推断目标人物的图像：这一阶段使用一个生成式对抗模型，训练模型学习从标准化后的姿态图形推断到目标人物图像。



03 代码细节

单帧关节数据接口：

```
[[239, 222, 0.889311894774437, 0, 0],
 [237, 253, 0.8638552576303482, 1, 1],
 [211, 251, 0.7791572511196136, 2, 2],
 [178, 259, 0.7320631295442581, 3, 3],
 [144, 246, 0.7404890656471252, 4, 4],
 [263, 255, 0.7786415293812752, 5, 5],
 [294, 280, 0.7076801657676697, 6, 6],
 [324, 292, 0.7676228582859039, 7, 7],
 [211, 339, 0.500228681601584, 8, 8],
 [203, 383, 0.37075577257201076, 9, 9],
 [199, 412, 0.577866718173027, 10, 10],
 [249, 342, 0.5103628300130367, 11, 11],
 [255, 389, 0.4914906499907374 4, 12, 12],
 [251, 439, 0.6989829614758492, 13, 13],
 [235, 216, 0.9053744971752167, 14, 14],
 [245, 218, 0.8780315220355988, 15, 15],
 [226, 219, 0.8289714753627777, 16, 16],
 [252, 221, 0.826052650809288, 17, 17]]
```

03 代码细节

浙江大学 ZHEJIANG UNIVERSITY

生成骨架代码：

```
import sys
sys.path.append(str(openpose_dir))
sys.path.append('./src/utils')


# openpose
from network.rtpose_vgg import get_model
# utils
from openpose_utils import create_label


weight_name = './src/PoseEstimation/network/weight/pose_model.pth'

model = get_model('vgg19')
model.load_state_dict(torch.load(weight_name))
model = torch.nn.DataParallel(model).cuda()
model.float()
model.eval()

'''make label images for pix2pix'''
test_label_dir = save_dir.joinpath('test_label_ori')
test_label_dir.mkdir(exist_ok=True)

total_img = input("please input total number of img: ")

for idx in range(0, int(total_img), 1):
    shape = np.load(str(Path('./data/skeleton/test_npy_ori/').joinpath('{:05}_shape.npy'.format(idx))))
    joint_list = np.load(str(Path('./data/skeleton/test_npy_ori/').joinpath('{:05}_joint_list.npy'.format(idx))))
    person_to_joint_assoc = np.load(str(Path('./data/skeleton/test_npy_ori/').joinpath('{:05}_person_to_joint_assoc.npy'.format(idx))))
    label, cord = create_label(shape, joint_list, person_to_joint_assoc)
    cv2.imwrite(str(test_label_dir.joinpath('{:05}.png'.format(idx))), label)
```

务求实学，存是去非

正其谊、不谋其利，明其道、不计其功

03 代码细节

在原有骨架代码中添加npy输出：

```
def get_pose(param, heatmaps, pafs):
    shape = heatmaps.shape[:2]
    # Bottom-up approach:
    # Step 1: find all joints in the image (organized by joint type: [0]=nose,
    # [1]=neck...)
    joint_list_per_joint_type = NMS(param, heatmaps)
    # joint_list is an unravel'd version of joint_list_per_joint, where we add
    # a 5th column to indicate the joint_type (0=nose, 1=neck...)
    joint_list = np.array([tuple(peak) + (joint_type,) for joint_type,
                           joint_peaks in enumerate(joint_list_per_joint_type) for peak in joint_peaks])

    global idx
    np.save(str(Path('./data/source/test_npy_ori/').joinpath('{:05}_joint_list.npy'.format(idx))), joint_list)##############################

    # Step 2: find which joints go together to form limbs (which wrists go
    # with which elbows)
    paf_upsamp = cv2.resize(pafs, shape, interpolation=cv2.INTER_CUBIC)
    connected_limbs = find_connected_joints(param, paf_upsamp, joint_list_per_joint_type)

    # Step 3: associate limbs that belong to the same person
    person_to_joint_assoc = group_limbs_of_same_person(connected_limbs, joint_list)

    np.save(str(Path('./data/source/test_npy_ori/').joinpath('{:05}_person_to_joint_assoc.npy'.format(idx))), person_to_joint_assoc##########

    idx += 1###############
    # (Step 4): plot results
    label,cord_list = create_label(shape, joint_list, person_to_joint_assoc)

    return label, cord_list
```

03 代码细节

浙江大学 ZHEJIANG UNIVERSITY

务求实学，存是去非

正其谊、不谋其利，明其道、不计其功

根据npy文件生成骨架图片：

```
os.environ['CUDA_VISIBLE_DEVICES'] = "0"
torch.multiprocessing.set_sharing_strategy('file_system')
torch.backends.cudnn.benchmark = True
torch.cuda.set_device(0)

save_dir = Path('./data/skeleton/')
save_dir.mkdir(exist_ok=True)

openpose_dir = Path('./src/PoseEstimation/')
sys.path.append(str(openpose_dir))
sys.path.append('./src/utils')



weight_name = './src/PoseEstimation/network/weight/pose_model.pth'

model = get_model('vgg19')
model.load_state_dict(torch.load(weight_name))
model = torch.nn.DataParallel(model).cuda()
model.float()
model.eval()

'''make label images for pix2pix'''
test_label_dir = save_dir.joinpath('test_label_ori')
test_label_dir.mkdir(exist_ok=True)

total_img = input("please input total number of img: ")

for idx in range(0, int(total_img), 1):
    shape = np.load(str(Path('./data/skeleton/test_npy_ori/').joinpath('{:05}_shape.npy'.format(idx))))
    joint_list = np.load(str(Path('./data/skeleton/test_npy_ori/').joinpath('{:05}_joint_list.npy'.format(idx))))
    person_to_joint_assoc = np.load(str(Path('./data/skeleton/test_npy_ori/').joinpath('{:05}_person_to_joint_assoc.npy'.format(idx))))
    label, cord = create_label(shape, joint_list, person_to_joint_assoc)
    cv2.imwrite(str(test_label_dir.joinpath('{:05}.png'.format(idx))), label)

torch.cuda.empty_cache()
```

03 代码细节

浙江大学 ZHEJIANG UNIVERSITY

务求实学，存是去非

正其谊、不谋其利，明其道、不计其功

生成中间帧：

```
'''
step为第n帧(1~29);
arr_s1为第一个npy数组;
arr_s2为第二个npy数组;
返回arr_d1数组;
'''
start = 0
end = 30
n = start - end + 1


arr_s1 = np.load(str(Path('./data/skeleton/test_npy_ori/').joinpath('{:05}_joint_list.npy'.format(start))))
arr_s2 = np.load(str(Path('./data/skeleton/test_npy_ori/').joinpath('{:05}_joint_list.npy'.format(end))))
arr_d1 = np.zeros((18, 5))
person_to_joint_assoc = np.load(str(Path('./data/skeleton/test_npy_ori/').joinpath('{:05}_person_to_joint_assoc.npy'.format(start))))



def process(step, arr_s1, arr_s2):
    k = step

    for i in range(18):
        arr_d1[i][0] = round(arr_s1[i][0] + (arr_s2[i][0] - arr_s1[i][0]) / n * (k + 1))
        arr_d1[i][1] = round(arr_s1[i][1] + (arr_s2[i][1] - arr_s1[i][1]) / n * (k + 1))
        arr_d1[i][2] = arr_s1[i][2] + (arr_s2[i][2] - arr_s1[i][2]) / n * (k + 1)
        arr_d1[i][3] = i
        arr_d1[i][4] = i
    return arr_d1


for i in range(n - 1):
    arr_d = process(i, arr_s1, arr_s2)
    np.save(str(Path('./data/skeleton/test_npy_ori/').joinpath('{:05}_joint_list.npy'.format(i+1))), arr_d)
    np.save(str(Path('./data/skeleton/test_npy_ori/').joinpath('{:05}_person_to_joint_assoc.npy'.format(i+1))), person_to_joint_assoc)
    np.save(str(Path('./data/skeleton/test_npy_ori/').joinpath('{:05}_shape.npy'.format(i+1))), [512, 512])
```

04

项目创新点

答辩人：吴家驹

04 项目创新点

务求实学，存是去非

修改骨架

本项目在原有的动作迁移模型的基础上，专注于武打动作的迁移，完成了一套武打动作骨架库的构建，并实现了武打动作的可控化生成。我们在动作迁移的过程中，将骨架信息保存在npy文件中，保存了18个关节点的位置信息，在项目中可以通过改变关节位置坐标，改变骨架的动作，然后就可以使用我们编写的代码，按照npy格式的骨架坐标文件生成新的骨架图片，骨架图片就可以在原本的模型中，迁移到目标视频当中。

生成新动作

本项目还提供了根据首尾两帧骨架，就能生成一个平滑的动作视频的功能，这样就辅助了新的武打动作的生成。不用把每一帧的关节信息都创建出来，只需要把关键帧的骨架信息构建出来，利用中间平滑补帧的功能，实现新的武打动作骨架的生成，这样就能在没有源视频的情况下，也能生成目标视频的武打动作生成。

正其谊、不谋其利，明其道、不计其功

05

项目成果应用与推广及社会效益

答辩人：吴家驹

05 可能的社会效益

影视

节省武打视频制作成本。使用动作迁移软件可以快速生成高质量的动作镜头，并提高武打视频制作效率，降低制作成本。

游戏

提升游戏体验。游戏开发者可以利用动作迁移技术生成更加逼真和具有流畅感的战斗动作，从而提升游戏的沉浸度，提高用户体验。

安全

安全培训和体验。使用动作迁移技术，在无风险的虚拟环境下，模拟人们遇到各种危险场景时的应对方式，从而提高安全培训和体验的效果。

文化

促进文化传承和发展。动作迁移可以记录流派传统武术动作、招式等，方便传道授业、传承更广，有助于推动武术文化的传承和发展。

浙江大学
ZHEJIANG UNIVERSITY

感谢观看