改进的 RetinaFace 复杂构件关键点检测算法

林 鑫 沈建新* 秦 顺 潘 峰
(南京航空航天大学机电学院 江苏 南京 210016)

摘 要 为了解决航天装备复杂构件进行喷涂作业时难以定位、缺少关键点数据集等问题,在三维建模软件上搭建构件模型,通过截图和标记关键点制作数据集,并针对数据集量少的问题采取数据增操作。研究现有的 RetinaFace 关键点检测算法并进行改进,将主干特征提取网络采用优化的 MobileNet 结构,学习率采用余弦退火衰减,算法输入、输出张量长度与不同构件对应的关键点数相一致。实验结果表明,模型迭代 500 轮后在验证集上的平均误差降至 0.062,能够有效地检测出待喷涂构件的关键点,性能优于同类算法。

关键词 航天设备 复杂构件 关键点检测 数据集 RetinaFace 余弦退火衰减

中图分类号 TP391.4 **文献标志码** A DOI:10.3969/j.issn.1000-386x.2024.08.045

A KEYPOINTS DETECTION ALGORITHM FOR COMPLEX COMPONENTS BASED ON IMPROVED RETINAFACE

Lin Xin Shen Jianxin* Qin Shun Pan Feng
(*College of Mechanical & Electrical Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, Jiangsu, China*)

Abstract In order to improve the problems of difficulty in positioning of spraying operation, and lack of datasets in keypoints detection of complex aerospace components, models were built in the 3D modeling software to make datasets of keypoints detection by taking screenshots and marking keypoints. Data augmentation methods were used in order to solve the problem of small sample size of datasets. On the basis of researching and improving existing RetinaFace keypoints detection algorithm, an optimized MobileNet structure was designed for the backbone feature extraction network and the learning rate was decayed by cosine warmup. The length of the input and output tensor was consistent with the number of keypoints corresponding to different components. The experimental results show that the average error on the validation set drops to 0.062 after 500 iterations of the model. The algorithm has better performance than similar algorithms, and can effectively identify the keypoints of components to be sprayed.

Keywords Aerospace equipments Complex components Keypoints detection Datasets RetinaFace Cosine decay with warmup

0 引 言

随着我国工业的不断发展,制造业所占的比重也越来越大。在工业生产中,喷涂是一个重要的加工过程。按照喷涂方式划分,喷涂作业可分为人工喷涂和机器人喷涂。人工喷涂劳动强度大,喷涂产品表面均匀性差[1],严重影响构件的可靠性,并且涂料中的化学物质对人体有害。近年来,机器人自动喷涂技术逐渐成为主流。

飞机起落架、涡轮、自动倾斜器等复杂结构件是航空航天设备的关键部件,对这些构件高质量的喷涂作业可以有效地提升航天器的性能。最新研制的航空航天装备上的复杂结构件形状复杂、成形精度难控制[2],使用现有喷涂技术对复杂构件进行喷涂时,普遍存在效率低下、定位不准确、浪费资源等问题,无法高效准

收稿日期:2021 - 02 - 16。国防重大项目培育基金项目(NP2020420)。**林鑫**,硕士生,主研领域:深度学习。**沈建新**,教授。**秦顺**,硕士生。**潘峰**,硕士生。

确地完成复杂构件表面的喷涂工作。随着航空航天设备的飞速发展,复杂结构件的制造逐渐趋向于小批量、多特征、多品种,这些特点就要求复杂结构件的喷涂作业具有很强的快速响应能力。

为了快速、准确地确定复杂构件的位置,检测出复杂构件关键点尤为重要。准确检测出复杂构件的关键点,便可以获得关键点在像素坐标系中的坐标值。利用坐标变换,使像素坐标系中的坐标值转换为世界坐标系中的坐标值,便能计算出复杂构件在空间中的位姿,精确地在三维空间中确定构件的坐标,从而为喷涂路径轨迹规划提供前期数据基础。

近年来深度学习技术不断发展,是机器学习领域中一个重要的研究方向。通过深度学习,机器人可以学到样本数据的内在规律,从而像人类一样进行分析和辨别,并可以智能化地对文字、图像[3-5]和声音[6]等数据进行处理。目前,深度学习关键点检测技术最主要的应用在于人脸关键点检测与人体姿态关键点检测,代表的算法有 DCNN(Deep Convolutional Network)[7]、TCDCN(Tasks Constrained Deep Convolutional Network)[8]、MTCNN(Multi-task Cascaded Convolutional Networks)[9]、TCNN(Tweaked Convolutional Neural Networks)[10]、RetinaFace[11]、CPN(Cascaded Pyramid Network)[12]等。考虑到本文构件关键点数选取为 5 点左右,而人体关键点检测算法中为了表示人体姿态,输出关键点数一般为 14 点、21 点,甚至更多。此类算法仅适用于人体肢体、躯干姿态检测等特定应用场合,若应用于构件关键点检测,将会破坏网络模型整体结构,无法达到检测效果。通过对各类算法的研究和分析,本文选择 RetinaFace 关键点检测算法进行优化和改进。RetinaFace 算法输出 5 个关键点,与构件关键点数接近,并且具有语义信息,即对应的关键点检测对应的位置,同时在检测过程中关键点输出顺序保持不变。本文所提出的构件关键点,对应的点只能检测对应位置处的特征,并且在数据制作过程中关键点按序生成,在输出时按同样的顺序输出,因此具有语义信息,可以保证输出结果的有效性。

本文首先在三维建模软件上制作构件三维模型,在各个角度截图并标注关键点制作数据集,针对数据集量过少的问题采取多种数据增强操作。将原算法中特征提取网络修改为轻量化的 MobileNet 结构,精简了网络参数,在不降低检测精度的前提下大量提升网络训练的速度,并且采用余弦退火衰减方法使学习率动态调整。同时根据不同构件具有不同的关键点数目,将原算法输入与输出张量的 5 个关键点修改为与不同构件的关键点数相一致,保证了结果的准确性。

1 数据集制作

1.1 关键点选取

一个复杂构件上有无数个点,选取什么样的点作为训练与检测的关键点尤为重要。拐点、角点、突起点、圆心点等特殊位置的点特征明显,在网络训练的过程中其特征便于深度神经网络的学习,应被选作为关键点。而一般平面上的点特征不明显,应尽量避免选择其为关键点。关键点在像素坐标系下的数值经过坐标变换便可转化为世界坐标系下数值,从而辅助确定构件的位姿。经过对本文选取的五种复杂构件特征的分析与对比,分别选取构件的关键点数为:自动倾斜器 5 点、发动机 4 点、机翼 4 点、涡轮 3 点、起落架 4 点。选取构件的关键点信息如图 1 所示。

(a) 自动倾斜器 (b) 发动机 (c) 机翼

(d) 涡轮 (e) 起落架

图 1 复杂构件图像以及关键点信息

1.2 实验数据集

由于航天设备构件关键点检测目前没有开源的数据集,因此首先在 Solidworks2017 三维建模软件上搭建三维模型,在各个角度截图制作图像数据集。再通过关键点标注软件进行手工关键点标注,生成关键点在像素坐标系中的坐标值。

不同的构件具有不同的关键点数量,关键点坐标值、回归框坐标值、置信度即为神经网络训练的标签值。将图像相对路径和标签打包生成 TXT 标签文件,完成关键点检测任务的数据集标签制作,数据集中样本量共有 1 662 个。

在 TXT 标签文件中,每两行组成一个构件图像的完整信息。第一行为构件图片所在目录下的相对路径,以"#"开头;第二行为构件的位置信息、关键点坐标值、置信度。前四个值为构件在像素坐标系中的左上角点坐标和构件的宽高,最后一个数值为构件在回

归框内的置信度,其余的数值是构件的关键点坐标值,以 0.0 作为分隔。部分构件的图像信息和标签值 TXT 文件如图 2 所示。

```
# photo/image0.jpg
107 143 459 337 324.0 190.0 0.0 325.0 390.0 0.0 325.0 455.0 0.0 0.98
# photo/image1.jpg
83 151 464 333 313.0 194.0 0.0 314.0 391.0 0.0 314.0 455.0 0.0 0.95
# photo/image2.jpg
80 147 463 350 311.0 194.0 0.0 312.0 389.0 0.0 312.0 453.0 0.0 0.87
# photo/image3.jpg
74 153 462 363 300.0 215.0 0.0 301.0 409.0 0.0 301.0 472.0 0.0 0.90
# photo/image4.jpg
106 163 441 342 324.0 212.0 0.0 325.0 404.0 0.0 325.0 467.0 0.0 0.88
# photo/image5.jpg
77 185 461 301 303.0 204.0 0.0 304.0 395.0 0.0 304.0 457.0 0.0 0.89
# photo/image6.jpg
77 155 461 310 311.0 194.0 0.0 312.0 383.0 0.0 312.0 445.0 0.0 0.89
# photo/image7.jpg
57 151 480 337 296.0 204.0 0.0 297.0 390.0 0.0 297.0 451.0 0.0 0.90
# photo/image8.jpg
75 208 463 315 298.0 243.0 0.0 298.0 439.0 0.0 298.0 503.0 0.0 0.87
# photo/image9.jpg
80 204 469 316 305.0 246.0 0.0 305.0 442.0 0.0 305.0 506.0 0.0 0.95
```

图 2　部分图像信息和标签值 TXT 文件

1.3　数据增强

数据增强是神经网络训练过程中常用的操作技巧,在数据集样本数量较小时用来防止过拟合。在神经网络搭建时,一定的深度可以获得更好的性能。然而随着神经网络的加深,网络中的参数也会逐渐增多。当数据集较小的时候,过多的参数会拟合数据集的所有特点,而非数据之间的共性,因此需要对数据进行数据增强。本文采取平移(Translate)、旋转(Rotate)、缩放(Scale)、仿射变换(Affine transform)、调整亮度和对比度(Adjust brightness and contrast)等方式对数据集进行增广,将数据由 1 662 幅图片扩充到 16 040 幅图片。同时,在数据增强的过程中,图像的回归框信息以及关键点信息随之发生对应的改变,保证了数据的有效性。数据增强后,五种构件的数据集样本量分别为发动机 3 200 幅(4 关键点)、自动倾斜器 3 200 幅(5 关键点)、机翼 3 200 幅(4 关键点)、起落架 3 200 幅(4 关键点)、涡轮 3 240 幅(3 关键点)。

每轮训练的过程中随机将 20% 的数据集划分为验证集,其余 80% 作为训练集,并将数据集随机打乱。随机划分与打乱使模型训练结果具有更好的鲁棒性,其算法如算法 1 所示。

算法 1　训练集与验证集划分和打乱

```
1. val_split = 0.2                          # 20% 的数据用于验证集
2. with open(annotation_path) as f:
3.     lines = f.readlines()
4.     np.random.seed(10101)                # 设置随机种子
5.     np.random.shuffle(lines)             # 随机打乱
6.     np.random.seed(None)
7. num_val = int(len(lines) * val_split)    #验证集样本数
8. num_train = len(lines) - num_val         #训练集样本数
```

2　改进的 RetinaFace 算法

2.1　对特征提取网络的改进

2.1.1　池化层

池化操作就是图像的尺度变换(Resize),通常采取的操作是最大值池化(MaxPooling)。最大值池化操作过程如图 3 所示。

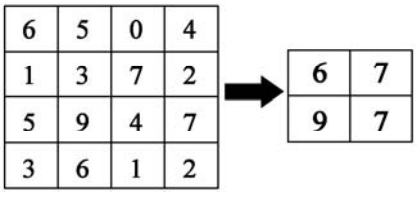


图 3　最大值池化操作过程

图 3 中输入为 4×4 的二维矩阵,池化操作块的尺寸为 2×2,池化操作步长为 2。池化的含义即将 2×2 池化操作块以 2 为步长滑过窗口,在每一块中将最大的数值保留作为新的输出,而舍去其余的数值,最终输出的尺寸为 2×2。根据图像特征的尺度不变性,最大值池化可以去除无关紧要的信息,图像中留下的则是物体关键的特征,是最能描述图像信息的特征。此外,最大值池化可以减小训练的参数,使网络训练所需要的时间更少。

2.1.2　Dropout 层

Dropout[13] 是神经网络训练过程中常用的训练技巧,是指按照一定的概率将某神经网络单元暂时从网络中中断。这种中断具有一定的随机性,通常将 Dropout 的概率值设置为 0.5 ~ 0.8,即某一个神经网络单元有 50% ~ 80% 的概率被舍弃,无法传入到下级的神经网络中,故而每一轮图像输入网络时,网络训练的参数都是不同的,有效地避免了过拟合。其示意图如图 4 所示。

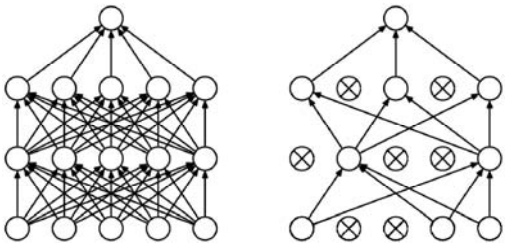


图 4　Dropout 示意图

2.1.3　激活函数

Leaky-ReLU 激活函数又称作“泄露修正线性单元”,其是在 ReLU 函数的基础上发展而来。ReLU 激活函数的表达式为:

$$f(x)=\begin{cases}x & x\geqslant 0\\ 0 & x<0\end{cases} \tag{1}$$

Leaky-ReLU 的表达式为：

$$f(x)=\begin{cases}x & x\geqslant 0\\ ax & x<0\end{cases} \tag{2}$$

式中：a 是(0,1)之间的任意数值。

通过式(1)、式(2)可知，ReLU 函数在输入 $x<10$ 时，无法起到激活作用，从而使该神经元“死亡”，导致梯度的消失。而 Leaky-ReLU 在 $x<0$ 的部分依然给定一个斜率值，达到了“泄漏”的目的，可对任意一个输入都起到激活的作用。

2.1.4 卷积层与卷积块

卷积的作用是提取特征，通过固定窗口大小的卷积块以固定的步长在图像上进行滑动，以窗口中的权重值与图像像素值相乘累加后的值作为输出结果。浅层的卷积可以提取出简单的特征，如图像中物体轮廓的边缘、拐点等，深层的卷积可以学习到更复杂的特征。

神经网络的卷积层并不是单独出现，而是与池化层(Pooling)、批归一化层(BatchNormalization)、激活函数层(Activation)、Dropoout 层等组成一个完整的卷积块。相比于单独的卷积层，卷积块可以对信息进行更好的处理。在网络代码的编排中，将卷积块整合成一个完整的函数，可以有效地对网络结构进行精简，避免大量重复的代码。本文卷积块算法函数如算法 2 所示。

算法 2 卷积块

```
def conv_block(inputs,conv_filters,strides=(1, 1))
                                        # 函数定义:卷积块
    x = DepthwiseConv2D((3,3), padding='same', strides=strides)(inputs)
    x = BatchNormalization()(x)
    x = Activation(Leaky_ReLU)(x)
    x = Conv2D(conv_filters,(1,1), padding='same', strides=(1,1))(x)
    x = Dropout(0.7)(x)
    x = BatchNormalization()(x)
    return Activation(Leaky_ReLU)(x)
```

2.1.5 改进的特征提取网络总结构

由 Deng 等提出的 RetinaFace 网络模型主要用于人脸关键点的检测，其主干特征提取网络采用 ResNet[14]。ResNet 结构的特点是通过短路机制加入了残差单元。残差单元实现了跳跃链接，缓解了神经网络中增加深度带来的梯度消失的问题。残差结构单元如图 5 所示。

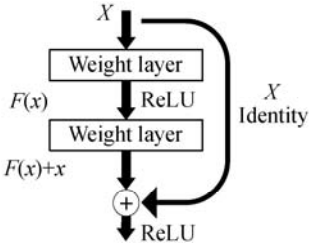


图 5 残差结构单元

ResNet 有多种结构，根据不同的深度进行划分，不同深度的 ResNet 结构如表 1 所示。

表 1 不同深度的 ResNet 结构

层号	输出尺寸	34-layer	50-layer	101-layer
1	112×112	7×7,64,stride=2		
2	56×56	3×3,max pool,stride=2		
		$\begin{bmatrix}3\times3,64\\3\times3,64\end{bmatrix}\times2$	$\begin{bmatrix}1\times1,64\\3\times3,64\\1\times1,256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1,64\\3\times3,64\\1\times1,256\end{bmatrix}\times3$
3	28×28	$\begin{bmatrix}3\times3,128\\3\times3,128\end{bmatrix}\times2$	$\begin{bmatrix}1\times1,128\\3\times3,128\\1\times1,512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1,64\\3\times3,64\\1\times1,256\end{bmatrix}\times3$
4	14×14	$\begin{bmatrix}3\times3,256\\3\times3,256\end{bmatrix}\times2$	$\begin{bmatrix}1\times1,256\\3\times3,256\\1\times1,1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1,256\\3\times3,256\\1\times1,1024\end{bmatrix}\times23$
5	7×7	$\begin{bmatrix}3\times3,512\\3\times3,512\end{bmatrix}\times2$	$\begin{bmatrix}1\times1,512\\3\times3,512\\1\times1,2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1,512\\3\times3,512\\1\times1,2048\end{bmatrix}\times3$
	1×1	average pool,100-d fc,softmax		
FLOPs		3.6×10^9	3.8×10^9	7.6×10^9

表 1 中，layer 为层数，stride 为滑动窗口步长，max pool 为最大值池化，average pool 为平均值池化，fc 为全连接层，softmax 为激活函数，FLOPs 为每秒所执行的浮点运算次数。可以看出，虽然 ResNet 网络特征提取效果好，但是网络结构复杂，参数量大，计算速度缓慢，同时对计算机的使用性能也有一定的要求。本文对 RetinaFace 网络模型进行改进，创新性地将其迁移应用至喷涂构件的关键点检测上。改进后的算法的主干特征提取网络将原 Resnet 结构修改为 MobileNet[15] 网络。在 MobileNetV1-0.25 网络的基础上进一步进行精简，将五个卷积块的卷积次数均改为两次，在不降低准确率的前提下，大大减少了网络的参数，同时也可以很好地提高检测的速度，达到实时性的要求。并将卷积层的激活函数改为 Leaky-ReLU，保证了函数在输入值为负数时依然可以起到激活的作用。改进的 MobileNet 主干特征提取网络结构框架如图 6 所示。

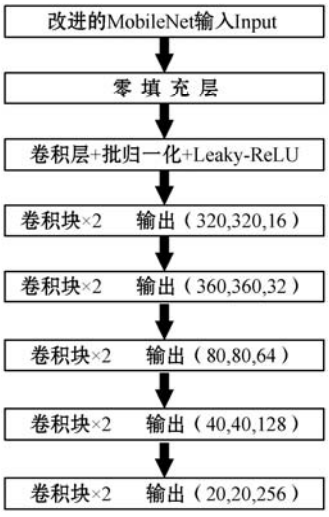


图 6　改进的 MobileNet 主干特征提取网络结构框架

2.2　对模型输入、输出张量的改进

在 RetinaFace 人脸关键点检测算法中,输入与输出张量长度为 15 ×1,其内容分别是:人脸框左上角点坐标(x,y);人脸的高度和宽度(h,w);脸部双眼、鼻子、嘴角五个关键点坐标(x_i,y_i,i = 1 ~5);框内存在人脸的置信度 c。由于不同的构件会具有不同的关键点数量,原算法中固定的输入、输出张量将不再适用,模型的输入与输出在关键点部分应与不同构件对应的关键点数相一致。具体改进算法如算法 3、算法 4 所示。

算法 3　标签数据输入函数

```
1. def Input_data(num_bbox = 4, num_keypoints, num_cls = 1):
                                        #定义标签数据输入函数
2. total_length = num_bbox + num_keypoints * 2 + num_cls
                                        #标签长度为 4 + 关键点数 ×2 +1
3. annotation = np. zeros((1, total_length))
4. for i in range(total_length)         #循环获取标签值
5.     annotation[i] = labels[i]
6. annotations = np. append(annotations,annotation,axis = 0)
                                        #数据整合
7. return annotations
```

算法 4　关键点输出张量

```
1. def Landmark (inputs, num_anchors = 2, num_keypoints):
                                        #定义输出函数
2.     outputs = Conv2D(num_anchors * num_keypoins * 2, kernel_
size = 1, strides = 1)(inputs)
3.     return Reshape([ -1,num_keypoints * 2])(outputs)
```

原算法中,在关键点部分输入与输出张量长度为固定值 5 ×2,与面部关键点数相一致。本文算法在函数构造时加入 num_keypoins 参数,可以在数据读取以及数据输出的时候,根据不同构件的不同关键点数改变输入与输出张量的长度,保证了检测结果的准确性。

2.3　对学习率的改进

在神经网络训练过程中,对学习率做出适当的调整十分重要。学习率的数值较大可以加快学习速率,帮助跳出局部最优值。缺点是会导致模型训练不收敛,并且单单使用大学习率容易导致模型不精确。学习率的数值较小可以帮助模型收敛,有助于模型细化,提高模型精度。缺点是过小的学习率会使模型收敛速度缓慢,并且可能导致无法跳出局部最优值。因此,大、小学习率的功能是互补的,合理设置学习率可以提升网络的性能。

本次训练学习率结合了大、小学习率二者的优点,采用余弦退火衰减法,先上升再下降。在上升过程中,学习率数值随训练的 Epoch 线性增长,模型训练速度逐渐加快。模型迭代 150 轮后,学习率达到最大值,开始逐渐衰减。在衰减过程中,呈 cos 函数下降。随着学习率数值的减少,模型可以得到很好的细化,逐渐达到最优解。学习率余弦退火衰减图像如图 7 所示。

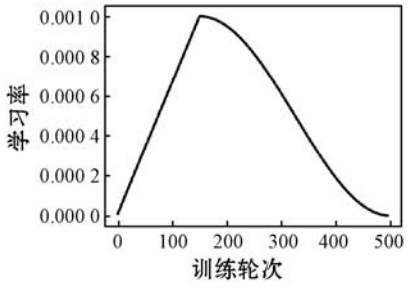


图 7　学习率余弦退火衰减图像

2.4　对损失函数的改进

在本文提出的关键点检测网络模型中,损失函数表达式如式(3)所示。

$$L = \lambda_1 L_{\text{cls}}(p_i, p_i^*) + \lambda_2 p_i^* L_{\text{box}}(l_i, l_i^*) + \lambda_3 p_i^* L_{\text{pts}}(l_i, l_i^*) \tag{3}$$

损失共包含三个部分:

1) 构件置信度损失 $L_{\text{cls}}(p_i, p_i^*)$,其值为该种类的预测结果的交叉熵损失,并采用 softmax 激活数进行激活,即 softmax 在二分类的情况的应用。其计算公式为:

$$L_{\text{cls}}(p_i, p_i^*) = p \times \lg p^* \tag{4}$$

2) 构件边框回归损失 $p_i^* L_{\text{box}}(l_i, l_i^*)$,其中 $l_i = \{l_x, l_y, l_w, l_h\}$,$l_i^* = \{l_x^*, l_y^*, l_w^*, l_h^*\}$,分别代表框内存在构件的条件下($p_i^*$)预测框和真实框(ground-truth box)的左上角点坐标以及宽、高值。边框回归损失为坐标采用 smooth-L_1 归一化后的对应的差值之和。

smooth-L_1 归一化能从两个方面限制梯度,即当预测框与真实框差别过大时,梯度值不至于过大;当预测框与真实框差别很小时,梯度值足够小。smooth-L_1 计

算公式为：

$$smooth_{L_1}(x)=\begin{cases}0.5x^2 & |x|<1\\ |x|-0.5 & \text{其他}\end{cases} \tag{5}$$

构件边框回归损失计算公式为：

$$L_{\text{box}}(l_i,l_i^*)=\sum_{i=x,y,h,w} smooth_{L_1}(l_i-l_i^*) \tag{6}$$

3）构件关键点回归损失 $p_i^* L_{\text{pts}}(l_i,l_i^*)$，其中 $l_i=\{l_{x1},l_{y1},\cdots,l_{xn},l_{yn}\}$，$l_i^*=\{l_{x1}^*,l_{y1}^*,\cdots,l_{xn}^*,l_{yn}^*\}$，分别代表框内存在构件的条件下（$p_i^*$）预测的 n 个构件关键点和基准点（ground-truth points）的坐标值，其中 n 根据不同构件的关键点数而改变。关键点回归损失为 x、y 坐标对应的差值采用 smooth-L_1 归一化后求和。关键点回归损失计算公式为：

$$L_{\text{pts}}(l_i,l_i^*)=\sum_{i=1}^{n} smooth_{L_1}(l_{xi}-l_{yi}^*) \tag{7}$$

此外，λ_1、λ_2、λ_3 分别为各部分损失的权重，表明某种损失值占总损失的比重。经过实验二证明，将 λ_1、λ_2、λ_3 分别设为 0.01、0.1、0.25 训练效果更好。在本文数据集制作过程中，确保了回归框内存在复杂构件的样本，因此置信度损失值权重应低于其余两值权重。即在训练过程中，文本加强了监控信号中更准确的回归框和关键点位置的重要性。而本文着重检测构件关键点，因此关键点定位准确性更为重要，其损失权重值应该更高。

3 实 验

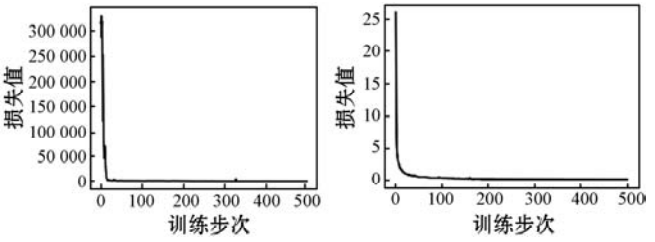
3.1 实验平台

本次训练平台使用实验室图像工作站 PC 机，CPU 型号为 i9-9900k，显卡型号为 RTX2080Ti，操作系统为 Windows 10 64 位。神经网络模型框架采用 Python-TensorFlow 1.13，使用 OpenCV 4.2 对图像进行数据增强，所有的样本尺寸均为 640×640。

3.2 实验结果

使用余弦退火衰减学习率对五种关键点检测模型各进行 500 次迭代后，模型逐渐收敛到一个较好的数值，损失值不再下降。模型在训练集上的平均损失降到 0.042，在验证集上的平均损失降至 0.062，模型有效收敛，能够很好地检测复杂构件的关键点。由于不同构件具有不同的关键点数量和特征信息，因此每种构件需分别训练模型权重，并对总损失求和后取平均值。训练过程中，关键点数量相同的构件使用相同结构的网络模型进行训练。训练结束后，将所有构件的权重模型放入模型库中并标号，在检测时首先需要获得构件类别的信息，根据种类选择相对应的模型权重来检测相匹配的构件，从而检测出不同构件的关键点。

关键点检测模型训练集和验证集上的平均损失值变化曲线如图 8 所示。



（a）训练集损失函数曲线 （b）验证集损失函数曲线

图 8 关键点检测损失值函数曲线

验证集上部分图像的关键点检测结果如图 9 所示。

图 9 验证集上部分图像关键点检测结果

3.3 算法评估

实验一 分别将未经过数据增强的数据集与经过平移、旋转、缩放、仿射变换等数据增强操作的数据集送入本文提出的网络模型进行训练。实验结果表明，虽然数据集数量小可以减少训练的时间，但是经过数据增强后的训练结果更稳定，对目标位置的变动、光强的变化都有很好的适应性，在验证集上的损失显著降低。实验结果如表 2 所示。

表 2 数据增强前后实验数据对比表

数据类型	本文算法	本文算法
数据集	原始数据集	数据增强
样本量	1 662	16 040
批量	40	
训练轮次	500	
学习率	余弦退火衰减	
训练时间/h	2.8	27.5
训练集损失	1.282	0.042
验证集损失	28.521	0.062

实验二 本次实验将损失函数中权重值 λ_1、λ_2、λ_3 分别取不同的数值，权重值表明某种损失值占总损

失的比重。本文算法主要目的是为了检测构件关键点,而构件是在其对应分类下进行模型训练,因此主要损失为边框回归损失以及关键点回归损失。将三种权重值取不同的数值进行训练后,实验结果证明,将 λ_1、λ_2、λ_3 分别设为 0.01、0.1、0.25 训练效果更好,其检测效果明显优于其他权重值。即在训练过程中,文本加强了监控信号中更准确的回归框和关键点位置的重要性,并且关键点更为重要。实验结果如表 3 所示。

表 3　不同损失值权重实验数据对比

数据类型	本文算法					
数据集	数据增强					
批量	40					
训练轮次	500					
损失函数权重	$\lambda_1=0.01$ $\lambda_2=0.1$ $\lambda_3=0.25$	$\lambda_1=0.01$ $\lambda_2=0.25$ $\lambda_3=0.1$	$\lambda_1=0.25$ $\lambda_2=0.1$ $\lambda_3=0.01$	$\lambda_1=0.25$ $\lambda_2=0.01$ $\lambda_3=0.1$	$\lambda_1=0.1$ $\lambda_2=0.25$ $\lambda_3=0.01$	$\lambda_1=0.1$ $\lambda_2=0.01$ $\lambda_3=0.25$
训练集损失	0.042	1.203	6.402	4.282	7.504	1.003
验证集损失	0.062	3.490	9.523	3.710	9.963	1.027

实验三　分别在训练的过程中采用恒定的学习率(learning_rate = 0.000 1)以及本文采用的学习率余弦退火衰减方法,对网络模型进行相同轮数的迭代。结果表明,恒定学习率虽然训练完成的时间更短,但在训练的损失值上,远远高于余弦退火衰减算法,训练效果不理想。实验结果如表 4 所示。

表 4　学习率改进前后实验数据对比表

数据模型	本文算法	
学习率	固定值 0.000 1	余弦退火衰减
数据集	数据增强	
批量	40	
训练轮次	500	
训练时间/h	22.5	27.5
训练集损失	5.563	0.042
验证集损失	8.238	0.062

实验四　本文将原算法中主干特征提取网络由 ResNet 改为优化的 MobileNet 网络,将各部分卷积次数改为两次,减少了网络训练的参数,并使用 Leaky_ReLU 作为激活函数。本实验将原 RetinaFace 算法与本文改进后的算法相对比,原算法网络结构复杂,参数多,训练时间长,而本文算法能在准确率相持平的前提下,有效缩短训练的时间,并提高检测的速度。实验测试结果如表 5 所示。

表 5　网络结构优化前后实验数据对比表

数据类型	原 RetinaFace 算法	本文算法
数据集	数据增强	
批量	40	
训练轮次	500	
学习率	余弦退火衰减	
训练时间/h	38.2	27.5
训练集损失	0.044	0.042
验证集损失	0.058	0.062

实验五　由于不同构件具有不同的关键点数,而现有的人脸关键点检测算法输出为 5 点或 68 点,无法形成对照实验。因此本实验数据集图像采用输出同为 5 点的自动倾斜器,将本文算法与现有的人脸关键点算法 DCNN、MTCNN、TCDCN 作对比,实验结果如表 5 所示,数据证明本文算法训练与检测结果优于同类算法,并且训练用时更短。

表 6　本文算法与同类算法实验数据对比表

数据类型	TCNN	MTCNN	TCDCN	本文算法
数据集	自动倾斜器数据部分			
样本量	3 200			
批量	40			
训练轮次	500			
学习率	余弦退火衰减			
训练时间/h	10	9	8	6
训练集损失	15.875	8.756	15.008	0.045
验证集损失	23.700	7.258	29.159	0.058

4　结　语

针对航空航天装备复杂构件进行喷涂作业时定位不精确或难以定位,以及喷涂构件关键点检测领域缺少数据集等问题,本文首先在三维建模软件上搭建构件的三维模型,在各个角度进行截图并且手工标记关键点,制作喷涂构件图像以及关键点数据集。同时研究现有的 RetinaFace 关键点检测算法并进行改进,改进后的模型主干特征提取网络采用优化的 MobileNet 结构,并将算法输入与输出张量长度与不同构件对应的关键点数相一致,训练过程中采用学习率余弦退火衰减等技巧,并经过实验 U 对比合理设置训练超参数。

实验结果表明,本文算法定位精度高,训练、检测速度快,性能优于同类算法。经过 500 轮迭代后,在训练集上的平均损失降至 0.042,在验证集上的平均损

失降至0.062,能够有效、精确地检测出待喷涂构件的关键点,为喷涂路径轨迹规划提供前期的数据基础。本文只选用了五种构件作为关键点检测的基础数据,后续的研究中,可以在数据集中添加更多的复杂构件,实现更多种类构件的关键点检测。

参 考 文 献

[1] 曹仁俊.喷涂机器人工作站的研究与实现[D].芜湖:安徽工程大学,2019.

[2] 王鹏程.复杂结构件加工特征用户自定义及识别方法[D].南京:南京航空航天大学,2016.

[3] Hinton G, Srivastava N, Krizhevsky A, et al. Improving neural networks by preventing co-adaptation of feature detectors[EB]. arXiv:1207.0580,2012.

[4] Krizhevsky A, Sutskever I, Hinton G. ImageNet classification with deep convolutional neural networks[J]. Communications of the ACM,2012,60(6):84 - 90.

[5] Cha Y, Choi W, Buyiikzturk O. Deep learning-based crack damage detection using convolutional neural networks[J]. Computer-Aided Civil and Infrastructure Engineering,2017,32(5):361 - 378.

[6] Mohamed A, Dahl G, Hinton G. Acoustic modeling using deep belief networks[J]. IEEE Transactions on Audio Speech and Language Processing,2011,20(1):14 - 22.

[7] Sun Y, Wang X, Tang X. Deep convolutional network cascade for facial point detection[C]//2013 IEEE Conference on Computer Vision and Pattern Recognition,2013:3476 - 3483.

[8] Zhang Z, Luo P, Loy C, et al. Facial landmark detection by deep multi-task learning[C]//European Conference on Computer Vision,2014:94 - 108.

[9] Zhang K, Zhang Z, Li Z, et al. Joint face detection and alignment using multitask cascaded convolutional networks[J]. IEEE Signal Processing Letters,2016,23(10):1499 - 1503.

[10] Wu Y, Hassner T, Kim K, et al. Facial landmark detection with tweaked convolutional neural networks[EB]. arXiv:1511.04031,2015.

[11] Deng J, Guo J, Zhou Y, et al. RetinaFace: Single-stage dense face localisation in the wild[EB]. arXiv:1905.00641,2019.

[12] Chen Y, Wang Z, Peng Y, et al. Cascaded pyramid network for multi-person pose estimation[EB]. arXiv:1711.07319,2018.

[13] Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: A simple way to prevent neural networks from overfitting[J]. Journal of Machine Learning Research,2014,15(1):1929 - 1958.

[14] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//2016 IEEE Conference on Computer Vision and Pattern Recognition,2016:770 - 778.

[15] Howard A, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[EB]. arXiv:1704.04861,2017.

(上接第 270 页)

[4] 杨婉香,严严,陈思,等.基于多尺度生成对抗网络的遮挡行人重识别方法[J].软件学报,2020,31(7):1943 - 1958.

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

[6] Criminisi A, Perez P, Toyama K. Object removal by exemplar-based inpainting[C]//IEEE Computer Society Conference on Computer Vision and Pattern Recognition,2003:2.

[7] Radford A, Metz L, Chintala S. Unsupervised representation learning with deep convolutional generative adversarial networks[EB]. arXiv:1511.06434,2015.

[8] Pathak D, Krahenbuhl P, Donahue J, et al. Context encoders: Feature learning by inpainting[EB]. arXiv:1604.07379,2016.

[9] Iizuka S, Simo-Serra E, Ishikawa H. Globally and locally consistent image completion[J]. ACM Transactions on Graphics,2017,36(4):1 - 14.

[10] Yu J H, Lin Z, Yang J M, et al. Generative image inpainting with contextual attention[EB]. arXiv:1801.07892,2018.

[11] He K M, Zhang X Y, Ren S Q, et al. Deep residual learning for image recognition[C]//IEEE Conference on Computer Vision and Pattern Recognition,2016:770 - 778.

[12] Zhang H, Goodfellow I, Metaxas D, et al. Self-attention generative adversarial networks[EB]. arXiv:1805.08318v1,2018.

[13] Goodfellow I J, Pouget-Abadie J, Mirza M, et al. Generative adversarial networks[EB]. arXiv:1406.2661,2104.

[14] Wang P Q, Chen P F, Yuan Y, et al. Understanding convolution for semantic segmentation[EB]. arXiv:1702.08502,2017.

[15] Gulrajani I, Ahmed F, Arjovsky M, et al. Improved training of wasserstein GANs[EB]. arXiv:1704.00028,2017.

[16] Arjovsky M, Chintala S, Bottou L. Wasserstein GAN[EB]. arXiv:1701.07875,2017.

[17] Liu Z W, Luo P, Wang X G, et al. Deep learning face attributes in the wild[EB]. arXiv:1411.7766v3,2015.

[18] Karras T, Aila T, Laine S, et al. Progressive growing of GANs for improved quality, stability, and variation[EB]. arXiv:1710.10196,2017.

[19] Zheng C X, Cham T J, Cai J F. Pluralistic image completion[EB]. arXiv:1903.04227,2019.