**High Dynamic Range Imaging**

**一丶绪论 Introduction**

* 1. **问题和动机**

普通相机由于动态范围有限，无法捕捉真实世界中无限范围的亮度信息，导致图像出现饱和区域，即部分区域过暗（欠曝）或过亮（过曝）。开发一种能够从低动态范围（LDR）图像中重建真实世界丢失亮度的方法，从而获得更逼真、更细节丰富的图像。

* 1. **背景资料和相关工作**

**1.2.1 HDR成像方法**

主要分为单曝光和多曝光两种方法。单曝光方法仅使用一张图像，而多曝光方法则融合多张不同曝光的图像。

**1.2.2 深度学习在HDR成像中的应用**

近年来，深度学习技术在HDR成像领域取得了显著进展，例如利用深度神经网络重建图像细节、融合图像、处理运动模糊等。

**1.2.3 现有方法的局限性**

现有的HDR成像方法主要关注图像的融合，而对图像中信息量丰富的区域提取关注较少。

* 1. **解决方法**

**1.3.1 预处理阶段**

利用Otsu方法对短曝光和长曝光图像进行分割，提取信息量丰富的区域。

**1.3.2 模型结构**

特征提取模块：从输入图像中提取特征。

视觉注意力模块：利用分割结果，提取图像中最可见的区域，并将其特征与输入图像特征进行融合。

空间对齐模块：对齐不同曝光图像的特征，避免运动模糊。

注意力模块：将不同曝光图像的特征与参考图像特征进行融合，并提取局部和全局特征。

重建模块：利用所有模块的输出，重建初始HDR图像。

优化模块：利用参考图像特征，对重建图像进行优化，提高图像质量。

**1.3.3 实验结果**

在NTIRE 2022 HDR成像挑战赛数据集上进行实验，结果表明，该方法在PSNR指标上优于大多数现有方法，并能够在保持低复杂度的同时，生成细节丰富的HDR图像。与HDR空间训练相比，在Sigmoid空间训练能够使模型更快收敛，并取得更好的性能。

**二丶相关工作 Related Work**

**2.1 前沿进展**

**2.1.1 深度学习模型的创新**

为了提升HDR成像的精度和效率，研究人员不断探索新的深度学习模型架构。例如，Transformer模型，最初用于自然语言处理，现在也被应用于图像处理领域。Transformer模型通过自注意力机制捕捉图像中的长距离依赖关系，从而生成质量更高的HDR图像。此外，多尺度网络能够在不同的尺度上处理图像信息，更好地捕捉图像的细节和全局结构。注意力机制则可以让模型关注于图像的特定部分，这对于HDR成像尤为重要，因为不同的区域可能需要不同的处理方式。

**2.1.2 运动模糊处理**

在多曝光图像中，由于拍摄过程中物体的运动，可能会出现运动模糊的问题。为了解决这个问题，研究人员提出了多种解决方案。例如，基于光流估计的方法可以估计图像中物体的运动方向和速度，从而对图像进行对齐。特征对齐的方法则通过匹配图像中的特征点来进行对齐。注意力引导的方法则利用注意力机制来指导图像对齐过程，从而减少运动模糊的影响。

**2.1.3 单曝光HDR成像**

除了传统的多曝光HDR成像方法，研究人员也开始探索单曝光HDR成像技术。例如，利用深度学习模型直接从一张LDR图像中重建HDR图像。这种方法可以避免多曝光图像融合过程中的复杂计算，提高成像效率。另外，也有研究尝试生成具有不同曝光的多张图像进行融合，从而得到HDR图像。

**2.1.4 低光照HDR成像**

在低光照环境下，由于光线不足，HDR成像面临更大的挑战。为了解决这个问题，研究人员提出了多种解决方案。例如，利用低光照图像增强技术，通过对图像进行去噪、增强亮度和对比度等处理，提高HDR图像的亮度和细节。另外，多帧图像融合技术也被应用于低光照HDR成像，通过融合多帧图像的信息，提高HDR图像的质量。

**2.2 尚未解决的问题**

**2.2.1 噪声处理**

在HDR成像的过程中，尤其是在图像分割、运动估计等步骤中，很容易引入噪声。这些噪声可能来自于传感器本身的噪声、图像处理算法的不完善，或者是环境因素导致的干扰。噪声的存在会降低图像的清晰度，影响最终成像的质量。因此，开发有效的去噪算法，以减少噪声对HDR成像的影响，是一个重要的研究方向。

**2.2.2 运动模糊**

尽管已有一些方法被提出来处理运动模糊问题，但当输入图像中存在较大的运动时，现有的技术仍然难以完全解决运动模糊。在多曝光HDR成像中，运动模糊会导致不同曝光图像之间的对齐问题，从而影响最终的合成效果。因此，研究更有效的运动检测和补偿算法，以提高HDR成像的质量，是一个具有挑战性的课题。

**2.2.3 单曝光HDR成像**

单曝光HDR成像技术试图通过从单个低动态范围(LDR)图像中恢复出高动态范围的细节，来简化HDR成像的过程。然而，这项技术目前仍处于研究阶段，其重建的精度和计算效率还有待进一步提高。如何利用深度学习等先进技术，从有限的图像信息中提取更多的动态范围，是单曝光HDR成像技术发展的关键。

**2.2.4 低光照HDR成像**

在低光照条件下，由于光线不足，图像的亮度和质量都会大幅下降。这给HDR成像带来了额外的挑战，例如如何有效地处理低光照图像中的噪声、如何恢复和保持图像的细节等。因此，研究适用于低光照环境的HDR成像技术，对于提高HDR成像的适用范围和实用性具有重要意义。

**2.2.5 实时HDR成像**

实时HDR成像要求在较短的时间内完成图像的捕获、处理和合成，这对于算法的效率提出了极高的要求。目前的HDR成像方法往往需要较长的处理时间，难以满足实时应用的需求，如视频拍摄、实时监控等。因此，研究更高效的算法和优化现有的技术，以实现实时HDR成像，是一个迫切需要解决的问题。

**2.3 未来研究方向**

**2.3.1 更有效的噪声处理方法**

为了提高HDR图像的最终质量，研究更有效的噪声处理方法是必要的。这包括探索基于深度学习的噪声去除技术，这些技术可以利用大量的数据来学习图像的内在特征，从而更有效地识别和去除噪声。此外，还可以研究新的去噪算法，如基于卷积神经网络(CNN)的方法，以及结合传统图像处理和深度学习的技术，以实现更好的去噪效果。

**2.3.2 更精确的运动估计方法**

运动估计的准确性对于减少运动模糊和提高HDR成像质量至关重要。未来的研究可以集中在开发基于Transformer的端到端运动估计网络，这些网络可以利用Transformer的自注意力机制来捕捉图像中的长期依赖关系，从而实现更精确的运动估计。此外，还可以探索结合传统光流方法和深度学习技术的混合模型，以提高运动估计的鲁棒性和准确性。**2.3.3 更先进的单曝光HDR成像技术**

单曝光HDR成像技术的进一步发展需要研究更先进的方法，例如基于多任务学习的方法，可以在单个网络中同时学习多个相关任务，如去噪、增强和HDR重建，以提高整体的成像效果。元学习（meta-learning）则可以用于快速适应新的成像环境和条件，从而提高单曝光HDR成像的适应性和效率。

**2.3.4 更高效的低光照HDR成像技术**

研究更高效的低光照HDR成像技术，例如基于图像增强、图像融合等方法，以提高低光照环境下的HDR图像质量。

**2.3.5 实时HDR成像算法**

研究实时HDR成像算法，例如基于轻量级网络架构、模型剪枝等方法，以满足实时应用的需求。

**三丶方法的具体细节 Details of the approach**

**3.1 工作概述**

我们的工作中首先提出了一种基于视觉注意力模块的HDR图像重建方法。该方法首先使用Otsu方法对短曝光和长曝光图像进行分割，提取包含更多细节的区域，并结合中曝光图像作为参考图像，输入到深度学习模型中进行HDR图像重建。模型结构包括特征提取模块、视觉注意力模块、空间对齐模块、注意力模块、重建模块和优化模块。实验结果表明，该方法在NTIRE HDR图像重建挑战数据集上取得了最佳性能，在PSNR指标上优于其他SOTA方法，同时在µ-PSNR、GMACs和参数数量方面也具有竞争力。该方法为HDR图像重建提供了一种新的思路，未来可以进一步研究如何在使用分割的同时避免噪声和错位问题，并探索更有效的图像分割方法和注意力机制，进一步提升模型性能。

**3.2 伪代码**

**3.2.1 utils.py**

augmentation(img, aug\_par): 图像增强函数，通过水平和垂直翻转来增强输入图像。

if aug\_par[0] == 1:

img = cv2.flip(img,0) #Vertical flip

if aug\_par[1] == 1:

img = cv2.flip(img,1) #Horizontal flip

return img

ev\_alignment(img, expo, gamma): 曝光值校正函数，用于校正图像的曝光值。

image\_read(img\_path, expo, gamma, train, img\_size, aug\_par): 读取图像并返回增强后的图像和对应的掩码。

mask[mask>0] = 1

mask = np.expand\_dims(mask,-1)

img\_new = ev\_alignment(img/255.0,expo,gamma)

final\_img = np.concatenate([img/255.0,img\_new],2).astype(np.float32)

final\_mask = np.concatenate([mask,mask],2)

path\_split(path, train, img\_size, aug\_par): 根据路径读取不同曝光值的图像并返回相应的

图像和掩码。

convert\_path(path): 将输入路径转换为地面实况路径。

HDR\_path = path.replace('LDR','HDR')

HDR\_path = HDR\_path.replace('\_short','\_gt')

HDR\_path = HDR\_path.replace('short','')

align\_path = HDR\_path.replace('gt.png','alignratio.npy')

imread\_uint16\_png(image\_path, alignratio\_path, aug\_par, train, img\_size): 读取图像并应用归一化处理。

# Load the align\_ratio variable and ensure is in np.float32 precision

align\_ratio = np.load(alignratio\_path).astype(np.float32)

# Load image without changing bit depth and normalize by align ratio

out=cv2.cvtColor(cv2.imread(image\_path,cv2.IMREAD\_UNCHANGED),cv2.COLOR\_BGR2RGB) / align\_ratio

if train:

out = cv2.resize(out,img\_size)

if aug\_par!=None:

out = augmentation(out,aug\_par)

return out

imwrite\_uint16\_png(image\_path, image, alignratio\_path): 将处理后的图像写入文件。

Create\_Dataset(batch\_size, input\_img\_paths, image\_size, stage, augment): 创建一个数据集，用于训练、验证或测试。

name\_sorting(path): 加载并排序图像路径。

psnr\_tanh\_norm\_mu\_tonemap(hdr\_nonlinear\_ref, hdr\_nonlinear\_res, percentile, gamma): 计算峰值信噪比（PSNR）。

# Added 000000000000001 to the denominator to avoid any zero division problem.

hdr\_nonlinear\_ref= tf.math.log(tf.cast(hdr\_nonlinear\_ref,tf.float64)/(1-tf.cast(hdr\_nonlinear\_ref,tf.float64)+.000000000000001))

hdr\_nonlinear\_ref= tf.clip\_by\_value(hdr\_nonlinear\_ref,0,tf.math.reduce\_max(hdr\_nonlinear\_ref))

hdr\_nonlinear\_res= tf.math.log(tf.cast(hdr\_nonlinear\_res,tf.float64)/(1-tf.cast(hdr\_nonlinear\_res,tf.float64+.000000000000001)))

hdr\_nonlinear\_res=

tf.clip\_by\_value(hdr\_nonlinear\_res,0,tf.math.reduce\_max(hdr\_nonlinear\_res))

hdr\_linear\_ref = hdr\_nonlinear\_ref\*\*gamma

hdr\_linear\_res = hdr\_nonlinear\_res\*\*gamma

norm\_perc = tfp.stats.percentile(hdr\_linear\_ref,percentile)

return psnr(tanh\_norm\_mu\_tonemap(hdr\_linear\_ref,norm\_perc), tanh\_norm\_mu\_tonemap(hdr\_linear\_res, norm\_perc))

peak\_signal\_noise\_ratio(y\_true, y\_pred): 计算PSNR。

# Added 000000000000001 to the denominator to avoid any zero division problem.

y\_true\_new= tf.math.log(tf.cast(y\_true,tf.float64)/(1-tf.cast(y\_true,tf.float64)+.000000000000001))

y\_true\_new = tf.clip\_by\_value(y\_true\_new,0,tf.math.reduce\_max(y\_true\_new))

y\_pred\_new= tf.math.log(tf.cast(y\_pred,tf.float64)/(1-tf.cast(y\_pred,tf.float64)+.000000000000001))

y\_pred\_new = tf.clip\_by\_value(y\_pred\_new,0,tf.math.reduce\_max(y\_pred\_new))

return tf.image.psnr(y\_pred\_new,y\_true\_new, max\_val=tf.math.reduce\_max(y\_true\_new))

tanh\_norm\_mu\_tonemap(hdr\_image, norm\_value, mu): 色调映射函数。

mu\_tonemap(hdr\_image, mu): 色调映射函数。

**3.2.2 model.py**

feature\_extractor(input\_img): 特征提取器，用于从输入图像中提取特征。

features = tf.keras.layers.SeparableConv2D(32,3,padding='same')(input\_img)

features\_max = tf.keras.layers.MaxPool2D(2)(features)

features\_avg = tf.keras.layers.AvgPool2D(2)(features)

concat = tf.keras.layers.concatenate([features\_max,features\_avg])

features=tf.keras.layers.SeparableConv2D(32,3,activation= 'relu',padding='same')(concat)

features = tf.keras.layers.UpSampling2D(2)(features)

return features

alingment(input\_i, input\_r): 对齐函数，用于对齐输入图像和参考图像。

input\_reference= tf.keras.layers.Conv2D(32,3,padding='same',activation='relu')(input\_r)

input\_reference1= tf.keras.layers.Conv2D(32,3,padding='same',activation='relu')(input\_reference)

input\_reference2= tf.keras.layers.Conv2D(32,3,padding='same',activation='relu')(input\_reference)

multiplication = tf.keras.layers.multiply([input\_i,input\_reference1])

out = tf.keras.layers.add([multiplication,input\_reference2])

return out

attention(input\_i, input\_r): 注意力机制，用于确定输入图像中的重要区域。

concat = tf.keras.layers.concatenate([input\_i,input\_r],axis=-1)

x = tf.keras.layers.SeparableConv2D(64,3,padding='same',activation='relu')(concat)

sigmoid = tf.keras.layers.SeparableConv2D(32,3,padding='same',activation='sigmoid')(x)

return sigmoid

alignment\_attention(input\_i, input\_r\_alingment, input\_r\_attention): 结合对齐和注意力机制的函数。

visual\_attention(inps, masks): 视觉注意力，用于增强图像的视觉特征。

reconstruction(inps, addition): 重构函数，用于从输入数据中重建图像。

reference = inps[:,:,:,-64:]

x = tf.keras.layers.SeparableConv2D(64,(3,3), padding = 'same')(inps) #(256,256)

x = tf.keras.layers.BatchNormalization()(x)

x = tf.keras.layers.Activation('relu')(x)

avg\_pool = tf.keras.layers.AveragePooling2D((2,2))(x) #(128,128)

max\_pool = tf.keras.layers.MaxPool2D((2,2))(x) #(128,128)

concat = tf.keras.layers.concatenate([avg\_pool,max\_pool])

......

# Making the ref image and add layer the same size of (128,128).

add\_avg\_pool = tf.keras.layers.AveragePooling2D((2,2))(addition)

med\_avg\_pool = tf.keras.layers.AveragePooling2D((2,2))(reference)

cat = tf.keras.layers.concatenate([x,add\_avg\_pool,med\_avg\_pool])

x = tf.keras.layers.SeparableConv2D(32,(3,3), padding = 'same', activation = 'relu')(cat)

x = tf.keras.layers.UpSampling2D((2,2))(x) #(256,256)

out = tf.keras.layers.SeparableConv2D(3,(1,1), padding='same',activation='relu')(x)

return (out)

refinement(input, reconstructed): 精细调整函数，用于优化重构图像。

visual\_mask\_expansion(mask): 扩展掩码通道数，用于增强掩码的表示能力。

model(): 定义整个模型，包括输入层、特征提取、对齐、注意力、重构和精细调整等部分。

**3.2.3 train.py**

导入所需的库: 导入TensorFlow、Keras和其他必要的库。

GPU检查: 检查是否有可用的GPU，并尝试设置GPU的内存增长。

读取模型: 从model.py文件中读取定义的模型。

模型配置: 配置模型，包括优化器、损失函数和评估指标。

数据目录: 定义训练和验证数据集的目录路径。

数据生成器: 使用Create\_Dataset函数创建训练和验证数据生成器。

加载权重: 如果存在，则加载权重。

训练模型: 使用数据生成器训练模型，并保存最佳权重和训练历史。

**3.2.4 test.py**

导入所需的库: 导入TensorFlow、Numpy和其他必要的库。

GPU检查: 检查是否有可用的GPU，并尝试设置GPU的内存增长。

读取模型: 从model.py文件中读取定义的模型。

模型配置: 配置模型，包括优化器、损失函数和评估指标。

测试数据目录: 定义测试数据集的目录路径。

数据生成器: 使用Create\_Dataset函数创建测试数据生成器。

保存预测结果: 将模型的预测结果保存为图像。

**3.3 相关公式**

**3.3.1 Otsu 方法计算阈值**

threshi = G(Yi)

其中，Yi 是 LDR 图像的亮度通道，G() 是 Otsu 函数，threshi 是图像 i 的阈值。

**3.3.2 特征提取**

Ci = concat (M(SepConv(Ii)), A(SepConv(Ii)))

Fi = Upsample(ReLU(SepConv(Ci)))

其中，M() 和 A() 分别是最大池化和平均池化函数，Ci 是拼接后的输出，Fi 是特征提取模块的输出。

**3.3.3 视觉注意力模块**

featuresL = F(multiply(maskL, IL))

featuresH = F(multiply(maskH, IH))

V = add(featuresL, featuresH)

其中，F 是特征提取函数，V 是 VAM 的输出特征。

**3.3.4 空间对齐模块**

Ref1 = ReLU(Conv(ref features))

Mi = multiply(ReLU(Conv(Ref1)), inp featuresi)

outi = add(ReLU(Conv(Ref1)), Mi)

其中，Ref1 是参考特征经过卷积和 ReLU 操作后的结果，Mi 是输入 LDR 特征经过卷积和 ReLU 操作后与 Ref1 相乘的结果，outi 是 Mi 与 Ref1 相加的结果。

**3.3.5 注意力模块**

Ri = ReLU(SepConv(concat(fi, fr)))

Si = Sigmoid(SepConv(Ri))

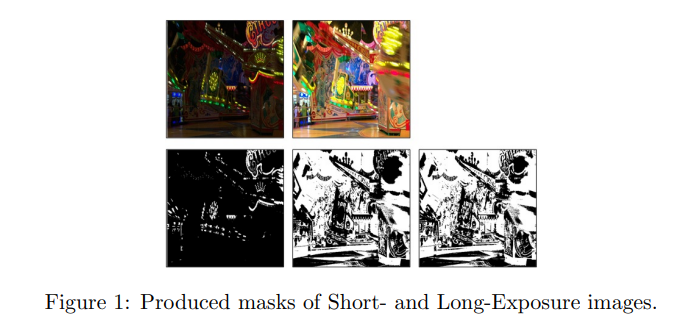
其中，fi 是伽马校正图像的特征，fr 是参考图像的特征，Si 是注意力模块的输出。

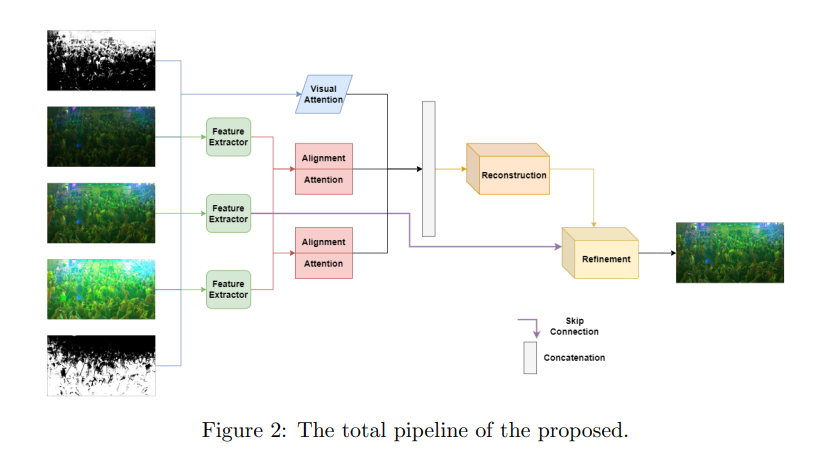
**3.3.6 HDR 空间逆 Sigmoid**

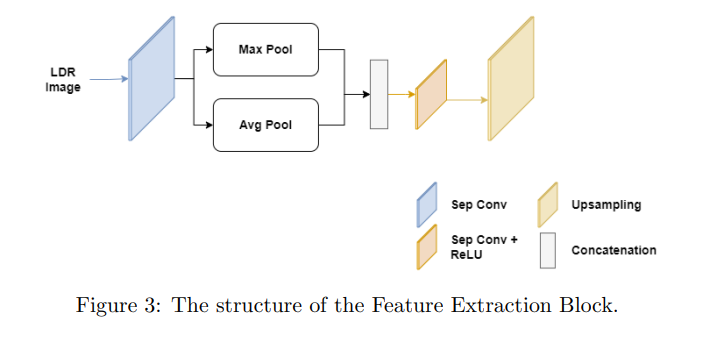
HDR = log(ˆ(y)1 − ˆy )

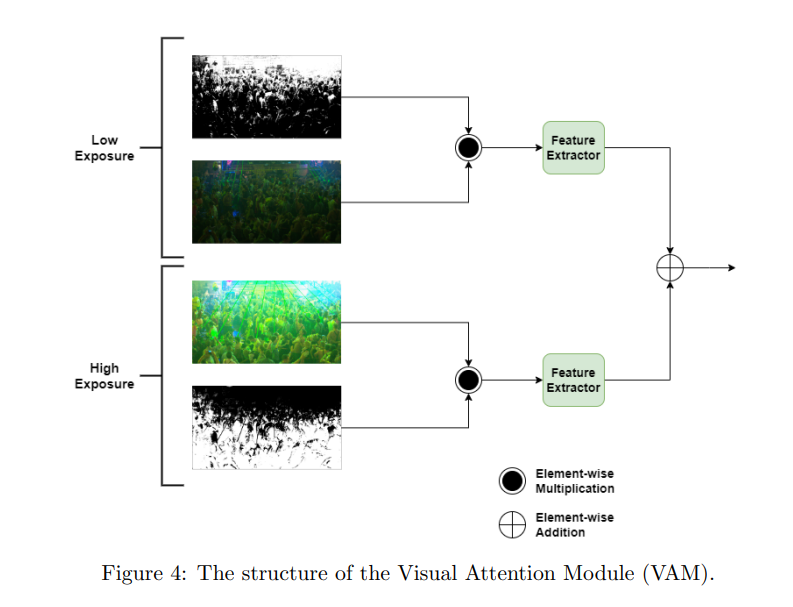
其中，HDR 是在 HDR 空间的输出，ˆy 是在 Sigmoid 域中的图像。

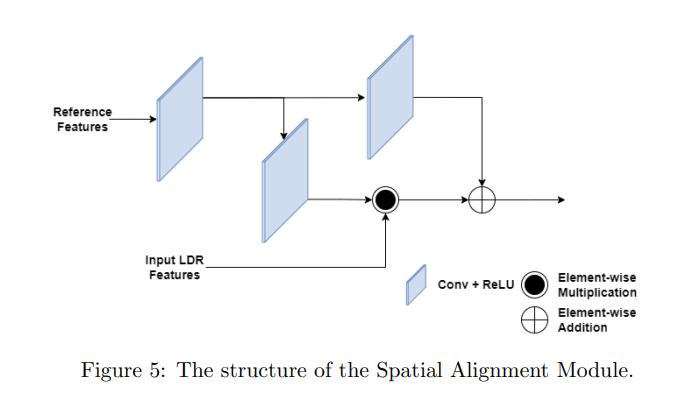
**3.4 相关图表**

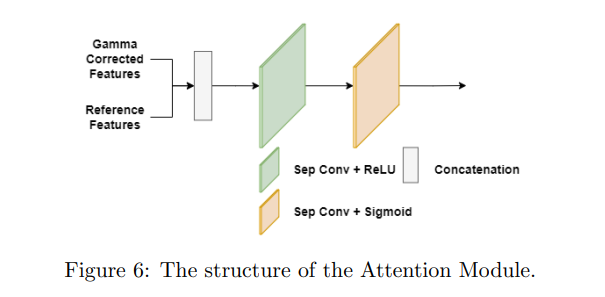


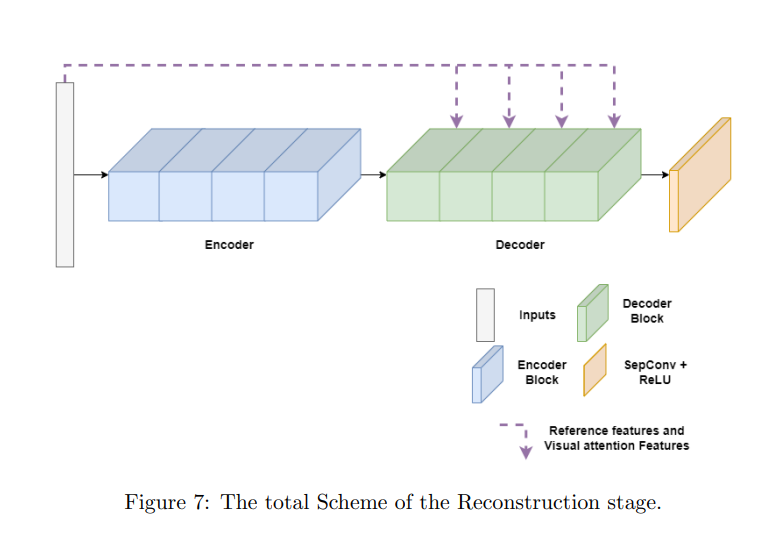


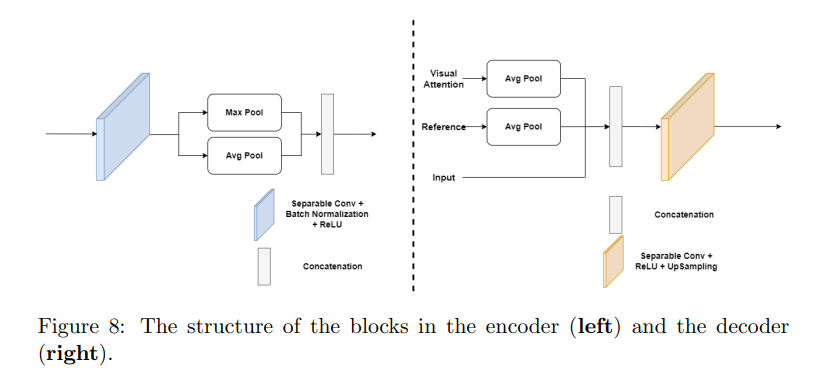


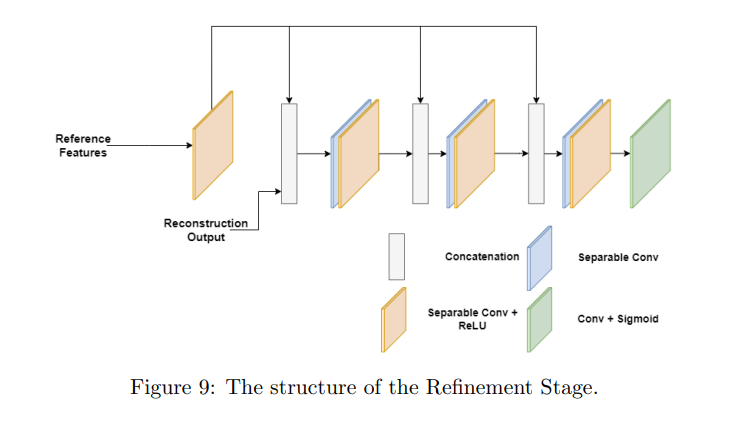












**四丶结果 Results**

**原图：**







**生成HDR图片：**



**五丶总结和讨论 Discussion and conclusions**

**5.1 主要收获**

视觉注意力模块有效: 通过将视觉注意力模块与图像分割相结合，模型能够有效地提取图像中包含更多细节的区域，并将其用于 HDR 重建，从而提高了重建图像的细节和清晰度。

图像分割有助于提高性能: 通过预先分割图像，模型能够专注于处理包含更多信息的区域，从而减少了计算量并提高了效率。

Sigmoid 空间更有利于模型训练: 将 Ground Truth 映射到 Sigmoid 空间后，模型能够更快地收敛并获得更好的性能。

模型性能优于现有方法: 在 PSNR 和 µ-PSNR 等指标上，所提出的方法优于大多数现有方法，证明了其有效性。

**5.2 思考**

噪声和错位问题: 当输入图像存在噪声或错位时，分割得到的区域可能会包含噪声或错位信息，从而导致重建图像出现噪声或错位。需要进一步研究如何避免或减少这些问题的影响。

其他分割方法: 可以探索其他图像分割方法，例如深度学习方法，来更精确地提取图像中包含更多细节的区域。

模型轻量化: 可以考虑模型轻量化技术，例如网络剪枝和知识蒸馏，来降低模型的复杂度和计算量，使其更适合在移动设备上部署。

实时性: 可以探索实时 HDR 成像技术，例如使用更快的网络结构或并行计算技术，来提高模型的运行速度。

**六丶个人贡献声明 Statement of individual contribution**

宋紫薇：数据库的收集改善以及实验的模拟运行再现

欧阳豪智：寻找合适实验项目，实验数据收集以及攥写项目报告

**七丶引用参考 References**

[1] S. Nayar and T. Mitsunaga, “High dynamic range imaging: spatially varying pixel exposures,” in Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662), vol. 1, pp. 472–479 vol.1, 2000.

[2] J. Tumblin, A. Agrawal, and R. Raskar, “Why i want a gradient camera,” in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05), vol. 1, pp. 103–110 vol. 1, 2005.

[3] M. McGuire, W. Matusik, H. Pffster, B. Chen, J. F. Hughes, and S. K. Nayar, “Optical splitting trees for high-precision monocular imaging,” IEEE Computer Graphics and Applications, vol. 27, no. 2, pp. 32–42, 2007.

[4] M. D. Tocci, C. Kiser, N. Tocci, and P. Sen, “A versatile hdr video production system,” ACM Trans. Graph., vol. 30, jul 2011.

[5] S. Hajisharif, J. Kronander, and J. Unger, “Adaptive dualiso hdr reconstruction,” EURASIP Journal on Image and Video Processing, vol. 2015, p. 41, Dec 2015.

[6] H. Zhao, B. Shi, C. Fernandez-Cull, S.-K. Yeung, and R. Raskar, “Unbounded high dynamic range photography using a modulo camera,” in 2015 IEEE International Conference on Computational Photography (ICCP), pp. 1–10, 2015.

[7] A. Serrano, F. Heide, D. Gutierrez, G. Wetzstein, and B. Masia, “Convolutional sparse coding for high dynamic range imaging,” in Proceedings of the 37th Annual Conference of the European Association for Computer Graphics, EG ’16, (Goslar, DEU), p. 153–163, Eurographics Association, 2016.

[8] G. Eilertsen, J. Kronander, G. Denes, R. K. Mantiuk, and J. Unger, “Hdr image reconstruction from a single exposure using deep cnns,” ACM Trans. Graph., vol. 36, nov 2017.

[9] L. She, M. Ye, S. Li, Y. Zhao, C. Zhu, and H. Wang, “Single-image hdr reconstruction by dual learning the camera imaging process,” Engineering Applications of Artiffcial Intelligence, vol. 120, p. 105947, 2023.

[10] P.-H. Le, Q. Le, R. Nguyen, and B.-S. Hua, “Single-image hdr reconstruction by multi-exposure generation,” in Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), January 2023.

[11] 张祖珩,陈晓冬,汪毅,等.基于掩膜Transformer的HDR图像重建算法[J/OL].激光与光电子学展:1-20[2024-06-19].http://kns.cnki.net/kcms/detail/31.1690.TN.20240612.0910.008.html.

[12] 付争方. 高动态范围图像信号恢复与增强算法研究[D].西安理工大学,2019.

[13] 方竞宇. 高动态范围彩色图像捕获与显示方法及技术研究[D].浙江大学,2017.

[14] 谢德红. 高动态范围图像显示再现技术的研究[D].武汉大学,2013.

[15] 于胜韬. 基于视觉感知的深度图超分辨和高动态范围视频压缩[D].西安电子科技大学,2019.

[16] 方华猛. 高动态范围图像快速合成与可视化理论及关键技术研究[D].武汉大学,2018.

[17] 余玛俐. 高灰度级图像的生成及多曝光融合技术研究[D].华中科技大学,2014.

[18]闫庆森. 高动态范围图像重建方法研究[D].西北工业大学,2022.DOI:10.27406/d.cnki.gxbgu.2019.000423.

[19]杨朋朋. 数字图像与视频的源取证技术研究[D].北京交通大学,2022.DOI:10.26944/d.cnki.gbfju.2021.000020.

[20] OpenCV4学习笔记（59）——高动态范围（HDR）成像https://blog.csdn.net/weixin\_45224869/article/details/105895367

[21] <https://github.com/rebeccaeexu/Awesome-High-Dynamic-Range-Imaging>

[22] <https://github.com/vivianhylee/high-dynamic-range-image>

[23] <https://blog.adobe.com/en/publish/2023/10/10/hdr-explained>