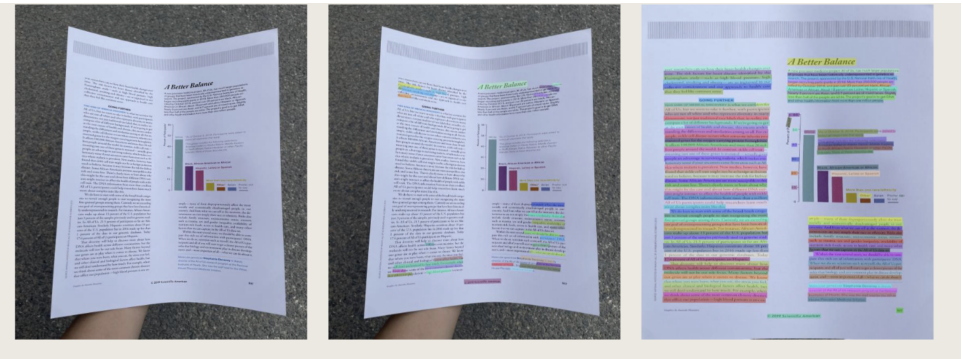
DocTr++文档矫正

> 目前最优的文档矫正方法，由中科院研发，第一篇对输入不限制的基于学习的文档矫正方法。
>
> 文档矫正可以用于各类用于拍摄角度、光照、文档实体畸形导致的扭曲文档情况，提升后续文档分析和OCR的准确度（能提升多少？从下图第三排看出还是可以大量提升检测）



motivation：

1. 现有方法需要输入的扭曲文档为整篇文档，输入只有局部文字时，矫正质量较差。

创新点：

1. 升级结构为层级的encoder-decoder，进行特征提取和分割
2. 调整了像素级别的映射关系：矫正前的扭曲图像（不做是否为整篇文档的限制）和矫正后的图像：当矫正后的图片像素源自矫正前的文档外区域，会被直接设置为一个零值，也就是矫正后的黑底。
3. 给出了一个现实世界的测试集和评价指标来验证矫正的质量。
    a. UDIR 训练集， Doc3D的拓展，其训练集包含10万张扭曲和矫正的图像；UDIR进一步裁减边缘
    b. UDIR 测试集，DocUNet Benchmark dataset的拓展，通过补充数据和裁减，保证三类情况各占1/3

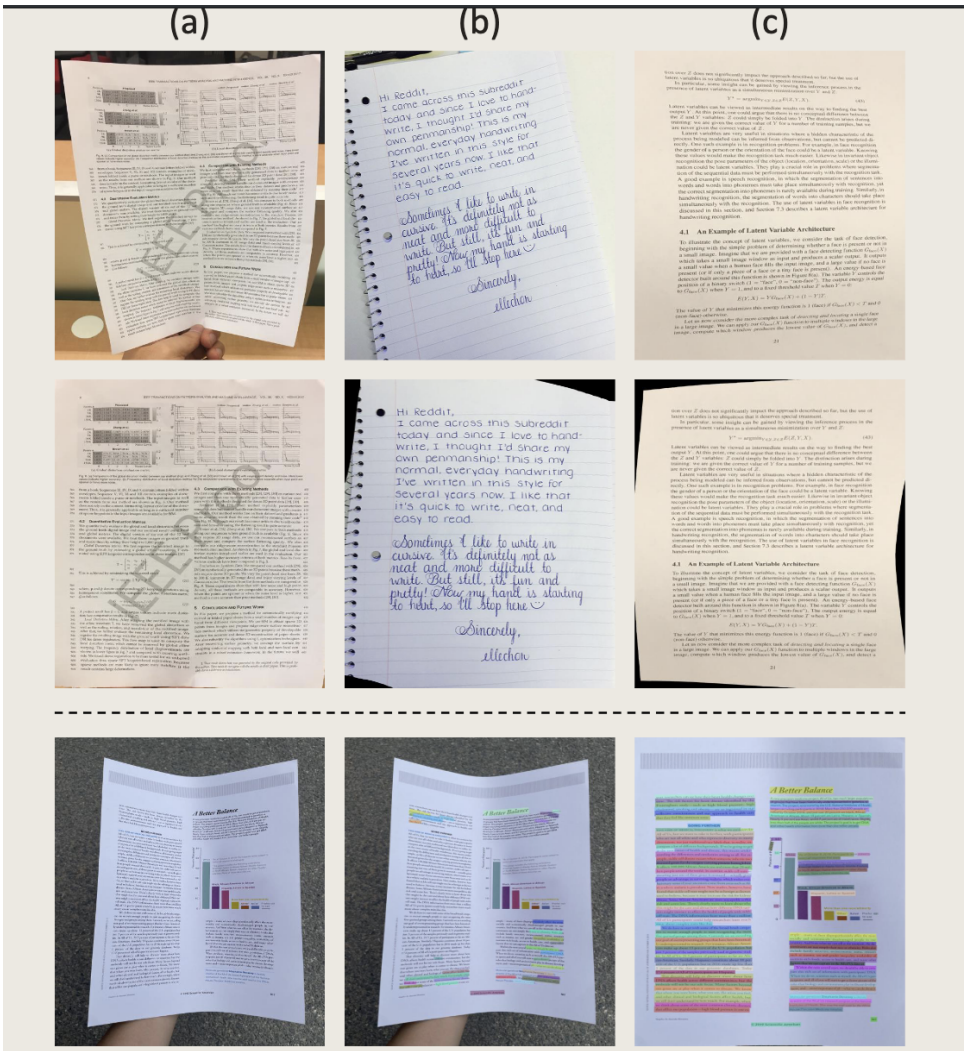


Fig. 1. Top row: three types of commonly distorted document images based on the presence of document boundaries: (a) w/ complete boundaries, (b) w/ partial boundaries, and (c) w/o any boundaries. Middle row: the rectified results of our method. Bottom row: the distorted image, the original detected texts, and the rectified one (highlighted), based on DBNet [31], [32].

三类不同的输入文档，是否有边界

现有方法

1. 3D重建：
    a. 建立被拍摄文档的3D表征，并将其映射到一个没有扭曲的平面。
    b. 需要拍摄多个平面/额外的激光扫描器的支持，不可拓展。
2. 提取变形表面参数的模型：
    a. 提取阴影、边界、文本线等参数用于矫正。
    b. 效果有限、计算量也不小。
3. 深度学习方法：网络学习像素位移场，重采样待矫正的图像，快速矫正
    a. 输入需要包含所有完整的边缘，否则矫正效果差（why？包含完整边缘的才有完整的角点和参考点信息）
    b. DocProj 将图像分割成多个patch，解决上述问题，但是计算复杂度大，且不能包含有背景的图像（上图3）。
    c. DDCP预估几个控制点和参考点+TPS差值算法来矫正。

d. PWUNet考虑本地patch的不同扭曲程度和提升全局的矫正效果。

网络结构

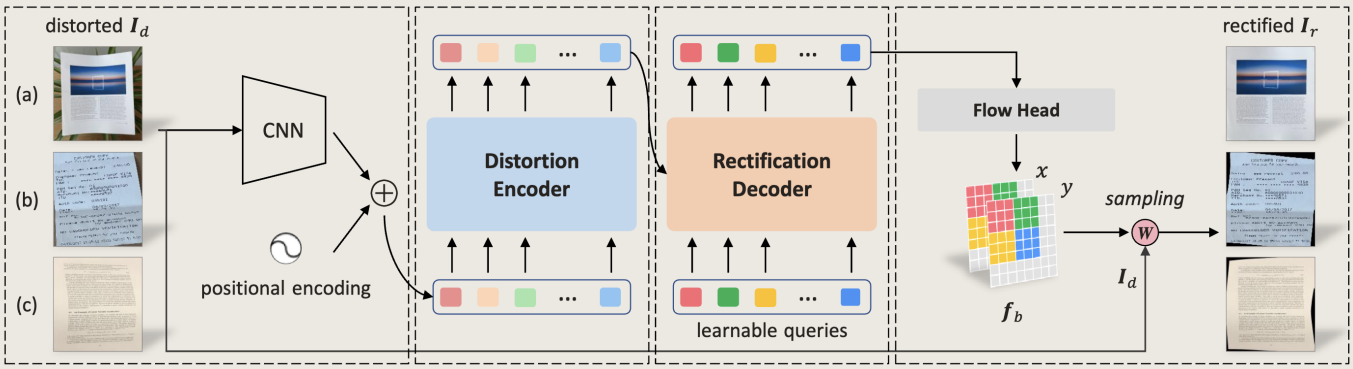


Fig. 2. An overview of our DocTr++ for unrestricted document image rectification. Given an arbitrary distorted document image I_d, we extract its features through a CNN backbone and a distortion encoder architecture. Then, the rectification decoder takes a fixed number of learned queries as input that attend to the encoder's output. These embeddings are parallelly transformed into per-patch warping flows f_b pointing to I_d. Finally, we use the predicted f_b to warp I_d and obtain the rectified image I_r through the bilinear sampling-based warping operation "W".

DocTr++网络结构

1. CNN提取2D特征，8倍下采样。
2. 扭曲编码器：学习结构信息，关注全局的弯曲文本线、纹理等。增加位置编码信息position embedding，因为transformer时扰动不变的。普通的transformer 配置
3. 序列化网络进一步进行序列关注编码。
   a. 编码解码器详细部分如下所示：

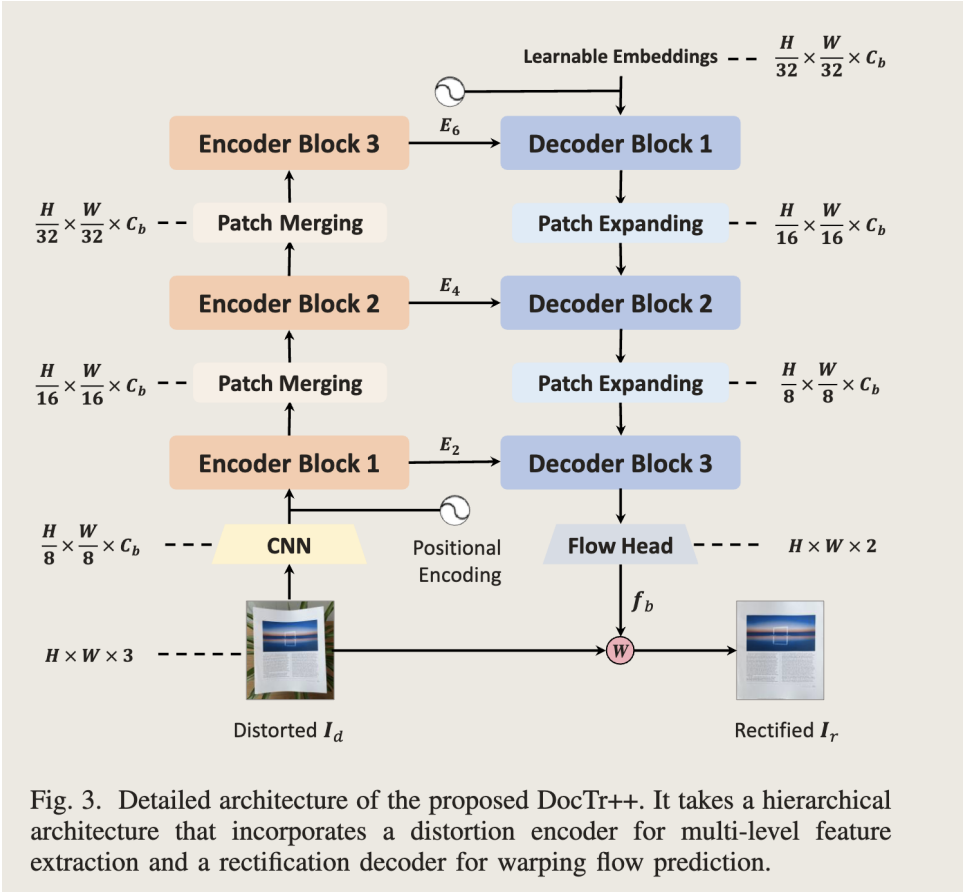


Fig. 3. Detailed architecture of the proposed DocTr++. It takes a hierarchical architecture that incorporates a distortion encoder for multi-level feature extraction and a rectification decoder for warping flow prediction.

4. 可学习的queries来提升对不同部分的关注度（相当于学习的关注向量，关注那一部分的特征重要），获得每个patch的warp flow。输出为相对原图八倍下采样。
5. flow head：精细化学习warp flow，进一步加权。两层卷积网络学习D6获得fm，另外有一个分支基于D6学习H/8 × W/8 × 8 × 8 × 9 的权重矩阵加权fm，H*W*2（垂直和水平的warp矩阵。）
6. 基于warp flow就可以对待矫正的图像进行重新矫正，矫正结果利用L1损失进行约束。

$$\boldsymbol{I}_r(u_0, v_0) = \boldsymbol{I}_d(\boldsymbol{f}_b^u(u_0, v_0), \boldsymbol{f}_b^v(u_0, v_0)),$$

$$\mathcal{L} = \|\boldsymbol{f}_{gt} - \boldsymbol{f}_b\|_1$$

代码分析：

1. 完整网络流程：

```
def forward(self, image1):
    fmap = self.fnet(image1)
    fmap = torch.relu(fmap)

    # fmap = self.TransEncoder(fmap)
    fmap1 = self.__getattr__(self.encoder_block[0])(fmap)
    fmap1d = self.__getattr__(self.down_layer[0])(fmap1)
    fmap2 = self.__getattr__(self.encoder_block[1])(fmap1d)
    fmap2d = self.__getattr__(self.down_layer[1])(fmap2)
    fmap3 = self.__getattr__(self.encoder_block[2])(fmap2d)

    query_embed0 = self.query_embed.weight.unsqueeze(1).repeat(1, fmap3.size(0), 1)
    fmap3d_ = self.__getattr__(self.decoder_block[0])(fmap3, query_embed0)
    fmap3du_ = self.__getattr__(self.up_layer[0])(fmap3d_).flatten(2).permute(2, 0, 1)
    fmap2d_ = self.__getattr__(self.decoder_block[1])(fmap2, fmap3du_)
    fmap2du_ = self.__getattr__(self.up_layer[1])(fmap2d_).flatten(2).permute(2, 0, 1)
    fmap_out = self.__getattr__(self.decoder_block[2])(fmap1, fmap2du_)

    # convex upsample baesd on fmap_out
    coodslar, coords0, coords1 = self.initialize_flow(image1)
    coords1 = coords1.detach()
    mask, coords1 = self.update_block(fmap_out, coords1)
    flow_up = self.upsample_flow(coords1 - coords0, mask)
    bm_up = coodslar + flow_up

    return bm_up
```

2. 编码模块：

```python
class TransEncoder(nn.Module):
    def __init__(self, num_attn_layers, hidden_dim=128):
        super(TransEncoder, self).__init__()
        attn_layer = attnLayer(hidden_dim)
        self.layers = _get_clones(attn_layer, num_attn_layers)
        self.position_embedding = build_position_encoding(hidden_dim)

    def forward(self, imgf):
        pos = self.position_embedding(torch.ones(imgf.shape[0], imgf.shape[2], imgf.shape[3]).bool().cuda())  # torch.Size([1, 128, 36, 36])
        bs, c, h, w = imgf.shape
        imgf = imgf.flatten(2).permute(2, 0, 1)
        pos = pos.flatten(2).permute(2, 0, 1)

        for layer in self.layers:
            imgf = layer(imgf, [imgf], pos=pos, memory_pos=[pos, pos])
        imgf = imgf.permute(1, 2, 0).reshape(bs, c, h, w)

        return imgf
```

3. 解码模块：

```python
class TransDecoder(nn.Module):
    def __init__(self, num_attn_layers, hidden_dim=128):
        super(TransDecoder, self).__init__()
        attn_layer = attnLayer(hidden_dim)
        self.layers = _get_clones(attn_layer, num_attn_layers)
        self.position_embedding = build_position_encoding(hidden_dim)

    def forward(self, imgf, query_embed):
        pos = self.position_embedding(torch.ones(imgf.shape[0], imgf.shape[2], imgf.shape[3]).bool().cuda())

        bs, c, h, w = imgf.shape
        imgf = imgf.flatten(2).permute(2, 0, 1)
        # query_embed = query_embed.unsqueeze(1).repeat(1, bs, 1)
        pos = pos.flatten(2).permute(2, 0, 1)

        for layer in self.layers:
            query_embed = layer(query_embed, [imgf], pos=pos, memory_pos=[pos, pos])
        query_embed = query_embed.permute(1, 2, 0).reshape(bs, c, h, w)

        return query_embed
```

效果展示：

TABLE II
QUANTITATIVE COMPARISONS WITH EXISTING METHODS ON THE PROPOSED UDIR TEST SET. "↑" INDICATES THE HIGHER THE BETTER AND "↓" MEANS THE OPPOSITE.

Methods	Venue	MSSIM-M ↑	LD-M ↓	ED ↓	CER ↓
Distorted	/	0.31	18.87	1181.47	0.3986
DocProj [48]	*TOG'19*	0.31	19.11	932.11	0.2995
DewarpNet [34]	*ICCV'19*	0.36	**17.91**	1037.04	0.3454
DocTr [36]	*MM'21*	**0.38**	18.80	**840.67**	**0.2988**
DDCP [40]	*ICDAR'21*	0.36	19.26	984.63	0.3324
DocGeoNet [44]	*ECCV'22*	**0.38**	18.60	872.69	0.3024
DocTr++	/	**0.45**	**12.47**	**666.49**	**0.2288**

TABLE I
QUANTITATIVE COMPARISONS WITH EXISTING METHODS ON THE DOCUNET BENCHMARK DATASET [39]. "↑" INDICATES THE HIGHER THE BETTER AND "↓" MEANS THE OPPOSITE.

Methods	Venue	MSSIM ↑	LD ↓	ED ↓	CER ↓
Distorted	/	0.25	20.51	1552.22	0.5089
DocUNet [39]	*CVPR'18*	0.41	14.19	1259.83	0.3966
DocProj [48]	*TOG'19*	0.29	18.01	1165.93	0.3818
FCN-based [41]	*DAS'20*	0.45	7.84	1031.40	0.3156
DewarpNet [34]	*ICCV'19*	0.47	8.39	525.45	0.2102
DocTr [36]	*MM'21*	**0.51**	7.76	464.83	0.1746
PWUNet [35]	*ICCV'21*	0.49	8.64	743.32	0.2623
DDCP [40]	*ICDAR'21*	0.47	8.99	745.35	0.2626
DocGeoNet [44]	*ECCV'22*	**0.50**	7.71	**379.00**	**0.1509**
Marior [47]	*MM'22*	0.48	**7.44**	593.80	0.2136
RDGR [37]	*CVPR'22*	**0.50**	8.51	**420.25**	**0.1559**
DocTr++	/	**0.51**	**7.52**	447.47	0.1695