



# 清华大学博士生暑期实践报告

## 欧卡智舶--水面无人驾驶清洁船贴边算法研究

作者：罗雁天

院系：清华大学电子工程系

时间：2021 年 8 月 5 日

学号：2018310742



# 目录

<b>1 背景</b>	<b>1</b>
1.1 智慧水务 . . . . .	1
1.2 水面清洁环卫与维护 . . . . .	2
1.3 现存问题与实践内容简述 . . . . .	3
<b>2 基于计算机视觉的水岸线检测算法</b>	<b>4</b>
2.1 基于图像语义分割的水岸线检测算法 . . . . .	4
2.1.1 Deeplabv3plus . . . . .	4
2.1.2 SETR . . . . .	5
2.1.3 Segformer . . . . .	7
2.1.4 实验结果对比 . . . . .	9
2.2 基于 lane detection 的水岸线检测算法 . . . . .	11
2.2.1 算法原理 . . . . .	12
2.2.2 数据标注 . . . . .	13
2.2.3 实验结果 . . . . .	14
<b>3 逆投影测距算法</b>	<b>15</b>
3.1 Adaptive IPM . . . . .	15
3.2 基于单应性矩阵的测距 . . . . .	18
<b>4 总结</b>	<b>21</b>
思想总结	22
参考文献	23

# 第一章 背景

## 1.1 智慧水务

目前我国流域水污染严重，水面垃圾、水草泛滥形势严峻，对于市容、水质产生巨大影响。十九大报告提出，实施重要生态系统保护和修复重大工程，优化生态安全屏障体系，构建生态廊道和生物多样性保护网络，提升生态系统质量和稳定性，加大生态系统保护力度。根据《全国 PPP 综合信息平台项目管理库 2020 年 1 月报》的数据统计显示，在管理库中生态建设和环境保护类 ppp 项目数为 926 个，占管理库项目总数的 9.8%；项目投资额为 10,059 亿元，其中水环境综合治理项目的占比超过 10%。据统计，2019 年度投资额在 20 亿以上的水环境治理项目，共计 22 个，总投资额累计约 1131.19 亿。全国生态环境建设市场已达万亿！

水环境综合治理属于典型的基础设施与公共服务领域，公益属性显著、投资大、专业性强、运营维护要求较高，是国务院、财政部、发改委相关政策中明确适宜和鼓励进行 PPP 运作的领域。伴随着水十条、河长制、湖长制具体出台落实，水环境治理开始迈入水域新基建，从单一走向综合、从被动治理走向主动防治，从传统维护走向智慧管理。良好的水域基建让城市生态变得更美丽，基建完成的水域如何科学、有效、可持续的进行维护成为首要任务和目标。十四五规划中，围绕水域将重点放在水域可持续管理和维护，水域新基建-智慧水务（图1.1）解决方案提上日程。

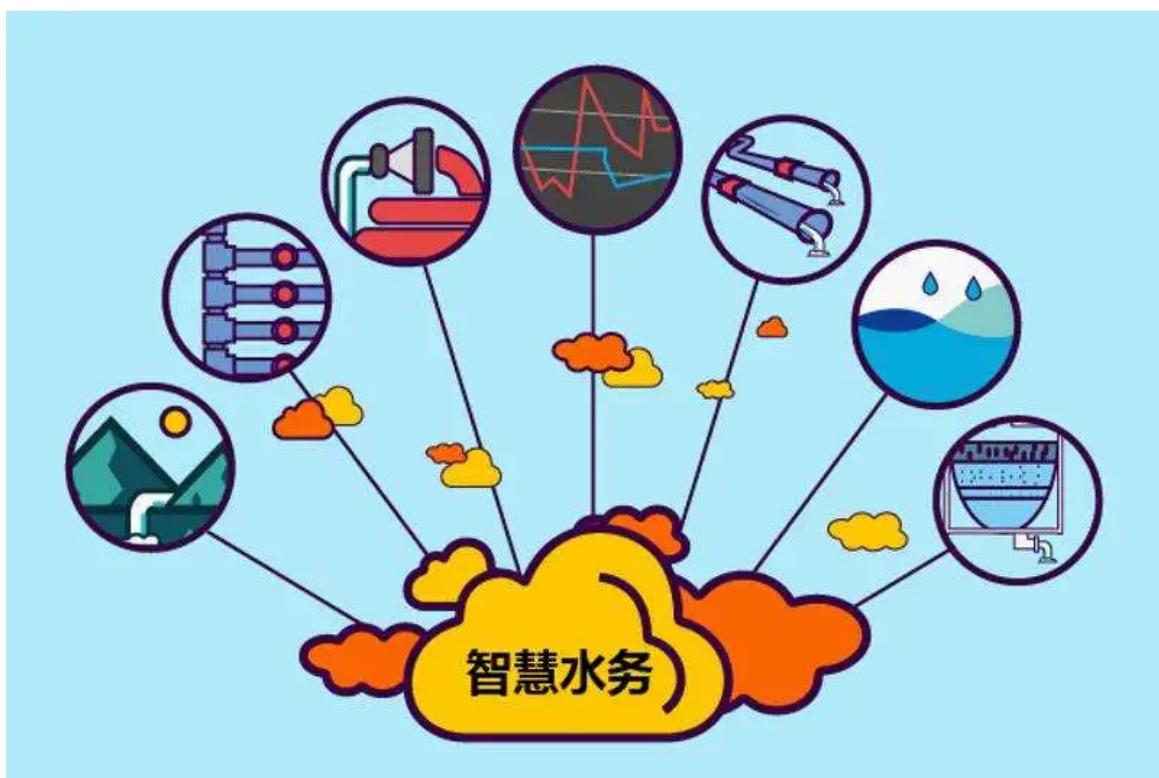


图 1.1: 智慧水务

智慧水务是以先进的信息科技（应用水务物联网数据采集、大数据 [11] 分析、模型仿真、移动化应用等）和先进的水领域专业技术为手段，以行业先进管理模式为标杆，以提升服务、运营能力、管理决策水平和绩效表现为 目标的智慧化平台。更通俗一点来说就是利用物联网、大数据、云平台等技术手段实现更好的水务信息化建设。

在当前如火如荼的智慧城市建设中，智慧水务是其中重要的组成部分。欧卡智能 [1] 依托水面无人驾驶技术，打造系列水面服务机器人实现高效的水域运维管理，同时结合公司在 AI 算法部分水面无人驾驶技术和机电一体化部分的整机/功能模块打造能力的丰富经验推出了系列的产品与解决方案：欧卡聚焦智慧水务管理平台下，用标准化水面无人驾驶服务机器人的方式进行水域管理与维护的三大解决方案：

- 水面清洁环卫与维护
- 水域数据收集与监测（水质、水深、水下地形）
- 水域安防巡检与排查（违法人员、活动、排污探测）

面向城市水域：湖泊，水库、内河；以及近海水域：港口、海岸线沿岸；逐步形成一套逐步形成一套从数据化监控、智能化维护到网格化预测的智慧水务整体解决方案，帮助水务企业打通企业内部乃至行业上下游的水域治理全套链条，实现对水域从监管、运营、服务等业务环节的高效管理，从而为智慧城市建设增添助力。

本次暑期实践主要关注“水面清洁环卫与维护”相关的内容，在此只介绍此部分的解决方案以及目前存在的问题。

## 1.2 水面清洁环卫与维护

水域清洁现有解决方案主要包括人工清理和打捞船清理。

 **人工清理** 依靠人工清理的方式，即采用半舱式或甲板机动驳船，由环卫工人手持网兜站在甲板上直接把垃圾捞上来，这种传统的打捞方法劳动强度大、工作环境恶劣，人身安全没有保证，清理垃圾效率低，且一般的机械化打捞设备作业面小，不能到达滩涂、狭窄河边工作，无法彻底清除垃圾。如图1.2a所示。



(a) 人工清理水域方式



(b) 打捞船清理水域方式

 **打捞船清理** 使用燃油驱动和人工驾驶的机械装置，但是体积较大，结构复杂且难以携带，不利于面积较小或形状复杂多变的城市河道、景观水池、饮用水库的垃圾清理，而且这些装置绝大多数使用燃油驱动，对清理水域会造成二次污染。如图1.2b所示。

考虑到以上问题，在湖泊、内河、水库的近海海域等水域生态区域，欧卡智能通过水面无人驾驶服务机器人平台，利用无线充电、高精度地图定位、垃圾识别等机电一体化技术，实现对水域漂浮垃圾的实时清理，不间断作业。通过水面无人驾驶技术，使得欧卡水面服务机器人在不需要遥控等人力干预下，即可自主完成清理作业，通过计算机、智能手持终端和物联网终端等设备，实现水质、垃圾等信息的实时查看、时空物联的远程精准控制。其相对于人工清理和大型水面清理装置的优势如图1.3所示。

### 解决方案对比

	智能清洁船	垃圾打捞割草一体船	人工清洁
运营成本	1000元/年	10000元/年	30000元/年/人
清理方式	无人自主清理	需要人员驾驶和打捞	需要大量劳动力人工清理
清理效率	每日覆盖50亩	每日覆盖50亩	每日覆盖10亩
清理时长	每日6-8小时	每日6小时	每日6小时
管理方式	数字无人化管理	需专人保养维护	人工管理
作业风险	无作业风险	人员因机械受伤、溺水风险	溺水风险
环境要求	无环境要求	只能在中大型水域使用	恶劣天气无法工作

图 1.3: 智能清洁船与人工清理、大型水面清理船的对比

## 1.3 现存问题与实践内容简述

欧卡无人驾驶船已经在城市内河、景观湖泊、人工水库、近海港口等大部分环境下较为准确的执行清洁任务，而对于岸边的清洁仍然有一定的缺陷。由于在湖泊、内河等地会有很多垃圾聚集在岸边，因此能够近距离的贴着岸边清洁是一个非常重要的挑战。目前公司主要是利用雷达和 GPS 来检测和重建小船当前的环境来进行贴边清洁，但是由于 GPS 定位有一定的误差，尤其对于湖面上方有树丛遮挡的场景，GPS 定位的误差会特别大，这也就导致了不能近距离的贴边带来的清洁效果不佳。

本次实践主要在计算机视觉的角度考虑，利用船头放置的摄像头拍摄当前场景，并结合图像语义分割等技术得到水岸线在摄像头视野中的位置，再利用逆投影的方式得到水岸线各个位置与摄像头之间的距离。实践相关代码已上传[https://github.com/luoyt14/orch\\_intern](https://github.com/luoyt14/orch_intern)，主要内容包括如下两方面：

- 基于计算机视觉的水岸线检测算法：包括 deeplabv3plus[2]、SETR[3]、Segformer[4] 等基于图像语义分割的算法，以及 Lane Detection[5] 的算法；
- 逆投影测距算法：一般来说，测距需要双目摄像头或者深度摄像头，单目无法测距，但是由于我们只需要水面上水岸线的点到摄像头的距离，拥有这些点都在同一个平面上的先验条件，使得单目相机测距成为可能。

## 第二章 基于计算机视觉的水岸线检测算法

本章介绍实践中使用的基于计算机视觉的水岸线检测算法，主要包括两个方面：

### 内容提要

- 基于图像语义分割的水岸线检测算法
- 基于 lane detection 的水岸线检测算法

## 2.1 基于图像语义分割的水岸线检测算法

本次实践中主要采用 Deeplabv3plus、SETR、Segformer 三种方法对船载摄像头拍摄到的图像进行分割，Deeplabv3plus 是一种经典的基于 encoder-decoder 结构和 CNN 网络的语义分割算法，SETR 是 CVPR2021 提出的基于 vision transformer[6] 结构的语义分割算法，Segformer 是最近刚提出的一种基于 Hierarchical Transformer Encoder 和纯 MLP decoder 的语义分割算法，具体细节以及实验结果在后文给出。

### 2.1.1 Deeplabv3plus

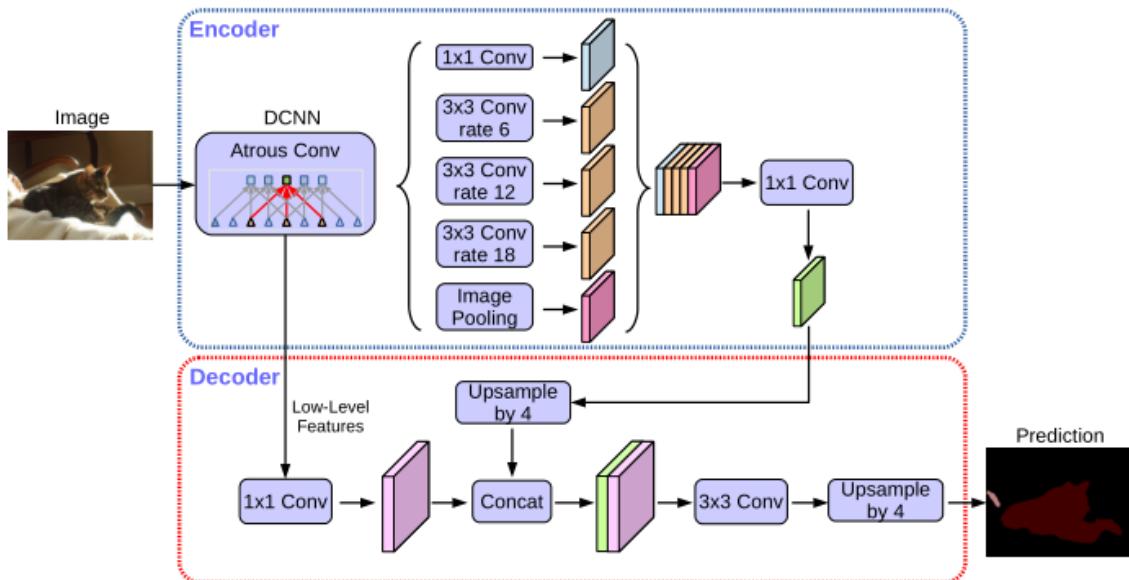


图 2.1: DeepLabv3plus 网络结构示意图

Deeplabv3plus[2] 采用 Encoder-Decoder 的架构，如图2.1所示，Encoder 部分，实际上就是 DeepLabV3[7] 网络。首先选一个低层级的 feature 用  $1 \times 1$  的卷积进行通道压缩（原本为 256 通道，或者 512 通道），目的是减少低层级的比重。由于编码器得到的 feature 具有更丰富的信息，所以编码器的 feature 有更高的比重，这样做有利于训练。对于 Decoder 部分，直接将编

码器的输出上采样 4 倍，使其分辨率和低层级的 feature 一致。将两种 feature 连接后，再进行一次  $3 \times 3$  的卷积（细化作用），然后再次上采样就得到了像素级的预测。

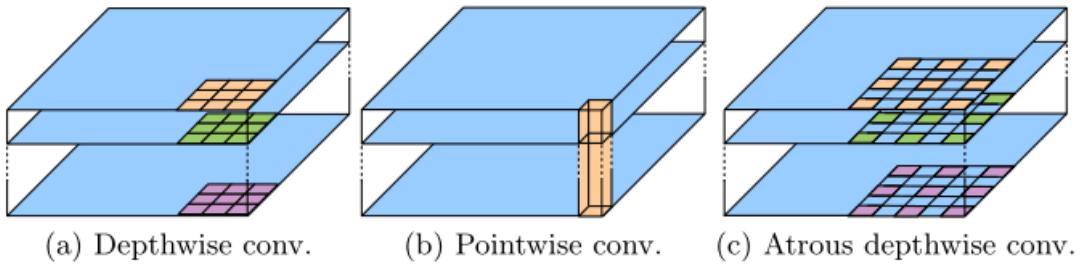


图 2.2: Depthwise 卷积、Pointwise 卷积以及空洞卷积示意图

其次网络中还用到了空洞卷积、可分离卷积来减小运算量，如图2.2所示，可分离卷积是将传统的卷积分解成 Depthwise conv 和 Pointwise conv 来减小计算量的，在 Deeplabv3plus 中，将空洞卷积用于 Depthwise conv 中，在保持 Deeplabv3 中感受野的同时，减小了运算量。对于二维输入  $x$ ，卷积核权重为  $w$ ，空洞卷积系数为  $r$  的位置  $i$  处的输出  $y$  可计算如下：

$$y[i] = \sum_k x[i + r \cdot k] w[k] \quad (2.1)$$

训练使用的 Loss 函数为交叉熵损失函数，计算如下：

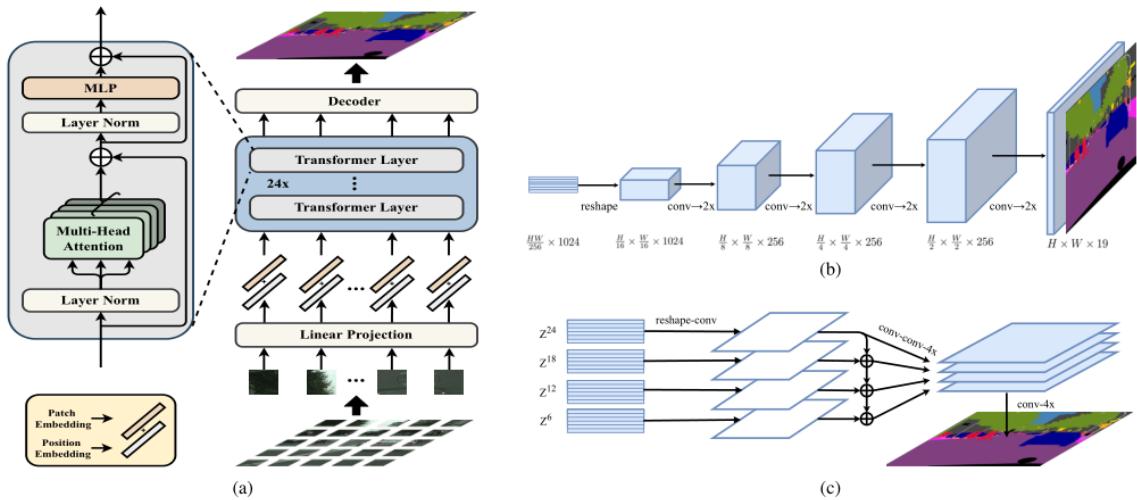
$$L = - \sum_h^H \sum_w^W \sum_c^C Y_{h,w,c} \log P_{h,w,c} \quad (2.2)$$

其中  $P \in \mathbb{R}^{H \times W \times C}$  是网络的输出， $Y \in \mathbb{R}^{H \times W \times C}$  是图像的 label 经过 one-hot 编码之后的张量， $H$  是图像的高， $W$  是图像的宽， $C$  是图像语义分割的类别数。

## 2.1.2 SETR

对语义分割而言，上下文 (context) 信息是提升性能最关键的因素，而感受野 (respect-field) 则大致决定了网络能够利用到多少的信息。通常，在编码器中，我们会在下采样的过程中逐层的降低空间分辨率，以减少计算资源的消耗同时有效的扩大了网络的感受野。然而，相关研究表明，网络的实际感受野远小于其理论感受野，过多的下采样操作会导致小目标的细节信息被严重损失甚至完全丢失。因此，如何既能够抽取全局的语义信息，又能尽量不损失分辨率，一直是语义分割的难点。传统基于 CNN 的语义分割框架从空洞卷积、注意力机制的方向进行优化，在一定程度上提升了语义分割的性能。

近几年来，Transformer[8] 和 self-attention 在自然语言处理领域取得了很好的成就，最近，也有很多人将 Transformer 用于图像识别领域，例如 ViT[6]、DeiT[9] 等。由于 Transformer 的一个特性便是能够保持输入和输出的空间分辨率不变，同时还能够有效的捕获全局的上下文信息，可以很好的解决语义分割面临的问题。因此 SETR 采用 Transformer 作为 Encoder 来抽取全局的语义信息 (整个过程不损失 image 分辨率)，代替传统 FCN 的编码部分，从序列-序列学习的角度，为语义分割问题提供了一种新的视角。



**图 2.3:** SETR 示意图: (a) 将图片切分成固定大小的 patches, 然后使用 Linearly embed 和位置编码, 然后将其输入标准的 Transformer 结构中得到编码后的特征, 为了能够进行逐像素的分割, 使用了两种不同的解码器, (b) 基于逐次上采样的解码器(将此种算法称为 SETR-PUP), (c) 基于多层特征融合的解码器(称为 SETR-MLA)

**图像转换为序列数据输入** SETR 的网络结构如图2.3所示, 假设输入图片  $x \in \mathbb{R}^{H \times W \times 3}$ , 按照  $16 \times 16$  的大小切分成  $\frac{H}{16} \times \frac{W}{16}$  个 patches, 对每一个 patch  $p \in \mathbb{R}^{16 \times 16 \times 3}$ , 使用一个线性映射函数  $f: p \rightarrow e \in \mathbb{R}^C$  得到 1D 的 embedding 向量。由于 Transformer 是无序的, 因此学习了一个特定的位置映射  $p_i$  来表征图像中每个 patch 的位置信息, 最后将  $p_i$  和  $e_i$  相加得到 Transformer 的输入序列数据  $E = \{e_1 + p_1, e_2 + p_2, \dots, e_L + p_L\} \in \mathbb{R}^{L \times C}$ , 在 SETR 中  $L = \frac{HW}{256}$ 。

**Transformer 编码器** 包括  $L_e$  层多头自注意力模块 (Multi-head Self-Attention, MSA) 和多层次感知机模块 (Multi-Layer Perceptron, MLP), 对于每一层  $l$ , 当前层的输入为上一层的输出为  $Z^{l-1} \in \mathbb{R}^{L \times C}$ (第一层的输入即为上一步得到的序列数据  $E$ ), 首先计算得到 Transformer Layer 输入的三元组 (query, key, value):

$$\text{query} = Z^{l-1}W_Q, \quad \text{key} = Z^{l-1}W_K, \quad \text{value} = Z^{l-1}W_V \quad (2.3)$$

其中  $W_Q, W_K, W_V \in \mathbb{R}^{C \times d}$  是三个线性层的参数,  $d$  是三元组 (query, key, value) 的维度。然后可以计算自注意力 (self-attention, SA) 如下:

$$SA(Z^{l-1}) = Z^{l-1} + \text{softmax}\left(\frac{Z^{l-1}W_Q(Z^{l-1}W_K)^T}{\sqrt{d}}\right)(Z^{l-1}W_V) \quad (2.4)$$

MSA 是将  $m$  个独立的 SA 连接起来, 通过一个线性层得到输出,  $MSA(Z^{l-1}) = [SA_1(Z^{l-1}); SA_2(Z^{l-1}); \dots]$ ; 其中  $W_O \in \mathbb{R}^{md \times C}$ , 实际应用中  $d$  一般设为  $C/m$ , 然后将 MSA 的输出过一个带跳跃连接的 MLP 模块, 得到本层的输出:

$$Z^l = MSA(Z^{l-1}) + MLP(MSA(Z^{l-1})) \in \mathbb{R}^{L \times C} \quad (2.5)$$

将各个 Transformer 层的输出记为  $\{Z^1, Z^2, \dots, Z^{L_e}\}$

**解码器** 将编码器得到的特征向量映射到原本图像大小来进行像素级的分割, 首先将编码器得到的特征向量进行 reshape:  $\frac{HW}{256} \times C \Rightarrow \frac{H}{16} \times \frac{W}{16} \times C$ , 然后通过解码器恢复到  $H \times W \times \text{categories}$ , 本文主要有三种解码器设计:

- **Naive upsampling:** 首先通过  $1 \times 1$  的卷积将最后一层 Transformer 层的输出  $Z^{L_e}$  的维度映射为分割类别数 (例如 cityscapes 19 类), 然后通过双线性插值 (大小  $\times 16$ ) 直接恢复到原图大小, 此种算法简称为 SETR-Naive;
- **Progressive UPsampling (PUP):** 如图2.3(b) 所示, 每次上采样到 2 倍大小, 上采样 4 次得到原图大小, 简称为 SETR-PUP;
- **Multi-Level feature Aggregation (MLA):** 如图2.3(c) 所示, 将不同 Transformer 层的特征拿出来, 首先 reshape 到  $\frac{H}{16} \times \frac{W}{16} \times C$  然后通过 3 个卷积层  $1 \times 1, 3 \times 3, 3 \times 3$ , 在第一个和第三个卷积层将  $C$  缩减到原来的一半。然后将得到向量上采样 4 倍并且从上到下将各高层的向量逐元素相加到低层, 再通过一个  $3 \times 3$  的卷积操作, 然后将不同层得到的向量堆叠, 最后双线性插值上采样 4 倍得到原图大小的解码输出, 简称为 SETR-MLA。训练使用的 Loss 函数依然是交叉熵损失函数(2.2), 除此之外, 还使用了 PSPNet[10] 中提到的辅助损失 (Auxiliary loss) 加速训练。

### 2.1.3 Segformer

虽然 SETR 已经有了足够好的效果, 但仍然有一定的缺点:

- 由于其使用 ViT 作为 Encoder, 参数量巨大, 不太适用于直接部署到船载摄像头;
- ViT 是柱状结构, 全程只能输出固定分辨率的 feature map, 比如  $1/16$ , 这么低的分辨率对于语义分割不太友好, 尤其是对轮廓等细节要求比较精细的场景;
- ViT 的柱状结构意味着一旦增大输入图片或者缩小 patch 大小, 计算量都会成平方级提高, 对显存的负担非常大;
- ViT 用的是固定分辨率的 positional embedding, 但是语义分割在测试的时候往往图片的分辨率不是固定的, 这时要么对 positional embedding 做双线性插值, 这会损害性能, 要么做固定分辨率的滑动窗口测试, 这样效率很低而且很不灵活。

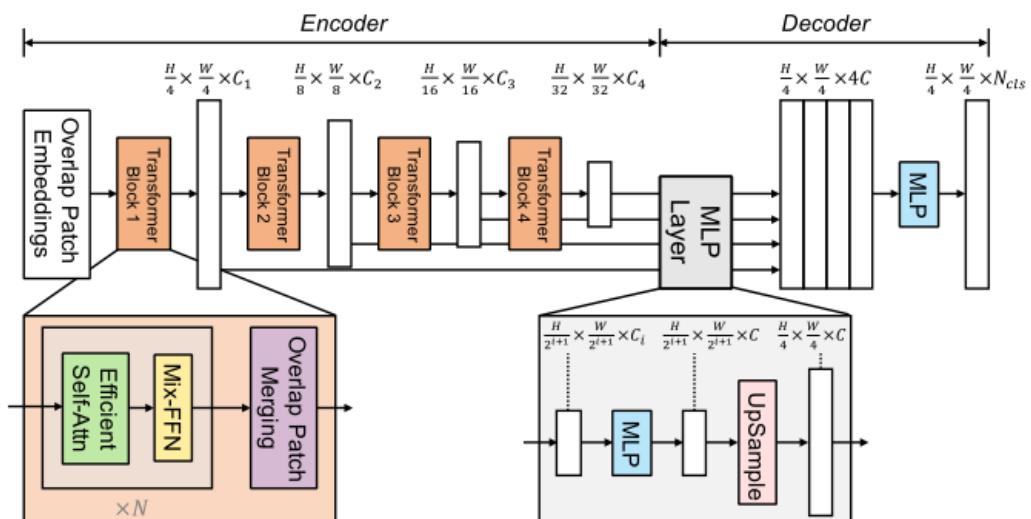


图 2.4: Segformer 示意图。主要包括两个模块, 1. 层次化的 Transformer 编码器来提取粗细粒度的特征; 2. 简单轻量级的 MLP 解码器融合不同层次的特征并预测语义分割 mask

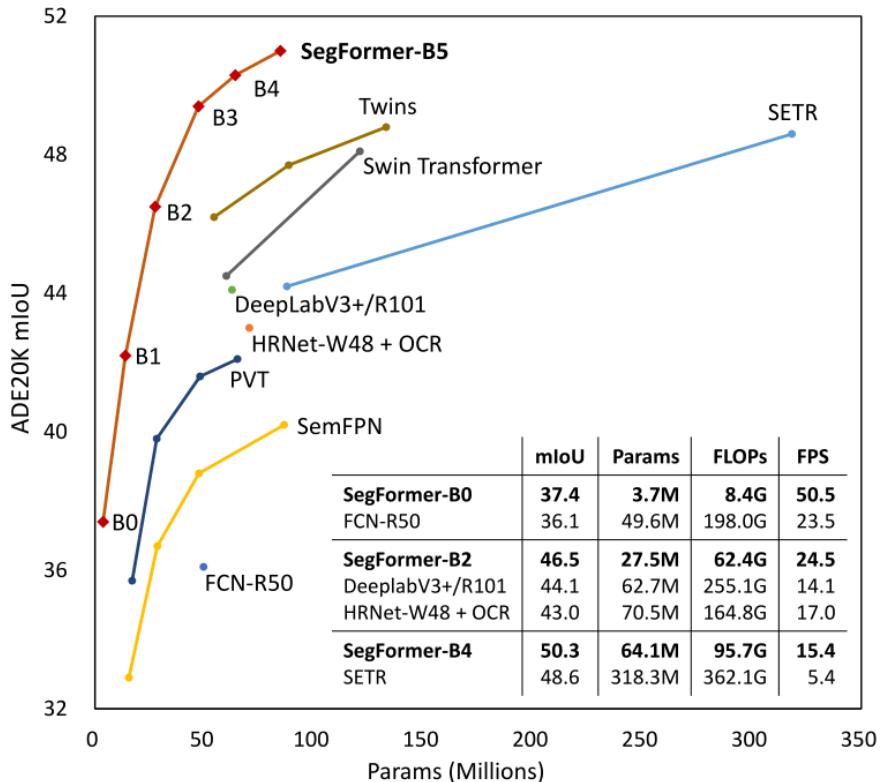


图 2.5: Segformer B0-B5 与其他算法在 ADE20k[12] 上的性能对比，横坐标表示参数量，纵坐标表示 mIoU

Segformer[4] 考虑到以上问题，重新设计了适用于语义分割的 Transformer Encoder，并且仅使用简单轻量级的 MLP 作为 Decoder，在不损失性能的前提下，极大的减小了模型的参数量大小，其网络结构如图2.4所示。



**Hierarchical Transformer Encoder** 主要包括如下几方面的改进：

- 之前 ViT 和 PVT 做 patch embedding 时，每个 patch 是独立的，这里对 patch 设计成有 overlap 的，这样可以保证局部连续性；
- 更高效的自注意力计算方法：原始的 Attention 计算方法为

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_{\text{head}}}}\right)V \quad (2.6)$$

复杂度为  $O(N^2)$ 。在此利用 [11] 中的方法进行以比例  $R$  进行维度缩减：

$$\begin{aligned} \hat{K} &= \text{Reshape}\left(\frac{N}{R}, C \cdot R\right)(K) \\ K &= \text{Linear}(C \cdot R, C)(\hat{K}) \end{aligned} \quad (2.7)$$

由此得到新的  $K \in \mathbb{R}^{\frac{N}{R} \times C}$ ，自注意力计算复杂度缩减到  $O\left(\frac{N^2}{R}\right)$

- 彻底去掉了 Positional Embedding，取而代之的是 Mix FFN，即在 feed forward network 中引入  $3 \times 3$  deepwise conv 传递位置信息，计算如下：

$$x_{\text{out}} = \text{MLP}(\text{GELU}(\text{Conv}_{3 \times 3}(\text{MLP}(x_{\text{in}})))) + x_{\text{in}} \quad (2.8)$$



**Lightweight All-MLP Decoder** 首先将 MiT 编码器不同层的特征  $F_i \in \mathbb{R}^{\frac{H}{2^{i+1}} \times \frac{W}{2^{i+1}} \times C_i}$  ( $i =$

1, 2, 3, 4) 通过一个 MLP 层统一维度, 然后上采样到原图  $1/4$  大小并 concatenate 起来, 然后再通过一个 MLP 层融合特征  $F$ , 最后再使用一个 MLP 层将融合后的特征预测分辨率为  $\frac{H}{4} \times \frac{W}{4} \times N_{cls}$  分割 mask, 计算如下:

$$\begin{aligned}\hat{F}_i &= \text{Linear}(C_i, C)(F_i), \forall i \\ \hat{F}_i &= \text{Upsample}(\frac{H}{4} \times \frac{W}{4})(\hat{F}_i), \forall i \\ F &= \text{Linear}(4C, C)(\text{Concat}(\hat{F}_i)), \forall i \\ M &= \text{Linear}(C, N_{cls})(F)\end{aligned}\tag{2.9}$$

训练使用的 Loss 函数依然是交叉熵损失函数(2.2)。针对不同的 overlap、Transformer 参数, 有 Segformer B0-B5 五种模型, B0 最快、参数量最小, B5 性能最好但是参数量较大, 但是都比 SETR 参数量小很多, 如图2.5所示。

## 2.1.4 实验结果对比

本小节将以上提到的 3 种算法在水面数据集上进行对比, 使用 python 语言以及 pytorch 框架进行编写, 在 Nvidia RTX 3090 显卡上进行实验, 考虑到参数量与计算复杂度的问题, Deeplabv3plus 使用了 ResNet18[13]作为 backbone, Deeplabv3plus 和 SETR 使用 pytorch 1.9.0+cuda 11.1+mmcv[14] 1.3.9+mmsegmentation[15] 0.15.0 环境进行实验, Segformer 使用 pytorch 1.8.0+cuda 11.1+mmcv[14] 1.2.7+mmsegmentation[15] 0.11.0 环境进行实验。SETR 使用了 SETR-PUP 用于实验, Segformer 使用了 Segformer-B5 用于实验。训练数据使用公司已经发布的数据集 USVInland Dataset[16], 分别在 USVInland Dataset 测试集以及使用小船采集的数据集上进行测试。



**训练集** 是在不同的城市内河、景观湖泊、人工水库、近海港口等采集并进行了像素级的标记, 只包括水和非水两个类别, 如图2.6所示, 标记图中红色的表示水面, 黑色的表示非水面的部分。在实际训练中选择了 1297 张图片进行训练, 图像包含有  $1280 \times 640$  和  $640 \times 320$  两种类型的分辨率, 在实际训练中, 我们将图像 resize 到  $512 \times 512$  进行训练。

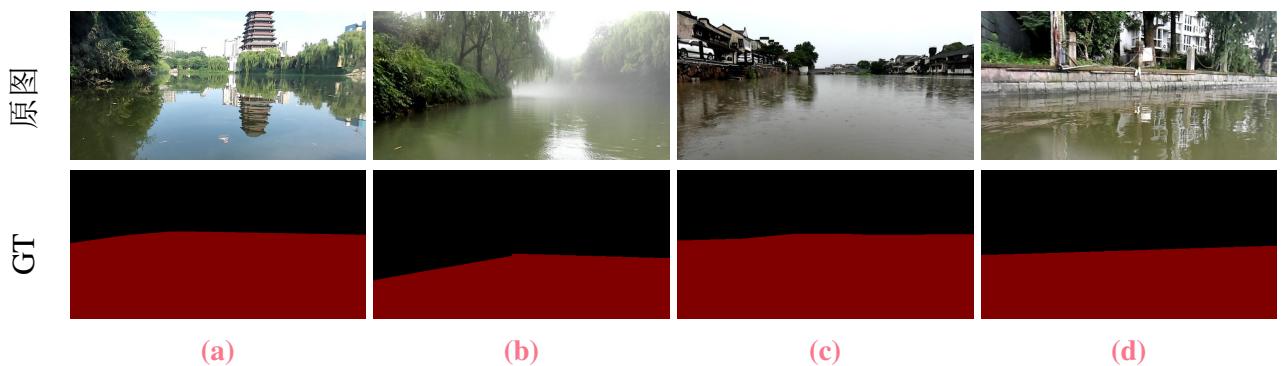


图 2.6: 训练集数据示例图, 第一行为原图, 第二行为语义分割标记图 (GT)



**USVInland Dataset 上进行测试与评估** 首先使用 USVInland Dataset 上剩下的数据作为测试集

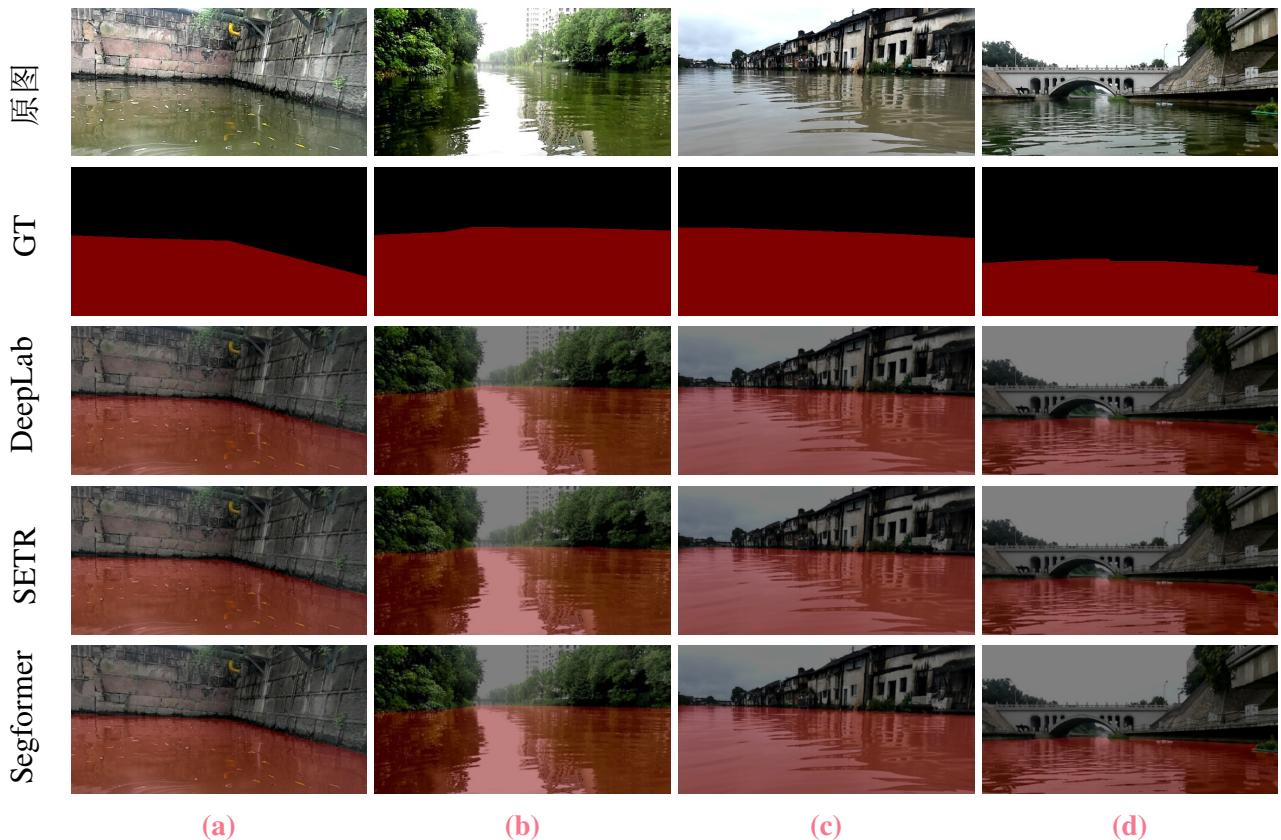
进行测试，并且采用 mIoU 来对 3 种不同的算法进行评估，IoU 的计算如下：

$$IoU = \frac{\text{target} \cap \text{prediction}}{\text{target} \cup \text{prediction}}$$

即每个类别预测区域和真实区域的交集与并集的比值，mIoU 为每个类别 IoU 的平均值。对于语义分割而言，mIoU 越大说明分割效果越好，三种算法 mIoU 的对比如表 2.1 所示，可以看出水面分割这个任务中，Segformer 的效果最好，但其实三种算法的精度都很高，基本都能正确的分割。可视化对比效果如图 2.7 所示，可以看出三种算法的分割效果都非常好，边缘也非常整齐。

**表 2.1:** Deeplabv3plus, SETR, Segformer 三种算法在 USVInland Dataset 上 mIoU 的对比

算法	deeplabv3plus	SETR	Segformer
mIoU	99.34	98.83	<b>99.39</b>



**图 2.7:** USVInland Dataset 测试集可视化对比图。第一行为测试集原图，第二行为语义分割标记图 (GT)，第三行为 Deeplabv3plus 算法的预测图，第四行为 SETR 算法的预测图，第五行为 Segformer 算法的预测图



**自己采集数据集上的测试** 从 USVInland Dataset 上的测试结果来看分割效果非常好，但由于训练集和测试集具有一定的相似性，因此有可能会有过拟合的现象。在此我们使用小船采集了西安外事学院鱼化湖的数据来做进一步的测试，这一批新采集的图像分辨率均为  $640 \times 480$ ，与训练集数据都不同。如图 2.8 所示，从第 4、5 行的对比图中可以看出，三种算法在大部分场

景分割效果都比较好。第 1 行对比图中，deeplabv3plus 在最左侧区域有误差，SETR 边缘不够光滑，Segformer 效果更好；第 2 行图同样 SETR 不够光滑，Segformer 和 deeplabv3plus 效果较好；第 3 行图，在摄像头部分被树叶遮挡时，deeplabv3plus 将树叶也分割到了水面类别中，属于分类错误，SETR 和 Segformer 都正确检测到了树叶，效果更好一些。综上所述，三种算法中 Segformer 表现更加鲁棒一些，是较好的一个算法。



图 2.8：西安外事学院鱼化湖场景三种算法对比图

## 2.2 基于 lane detection 的水岸线检测算法

对于无人船贴边清洁而言，只需要检测到小船左侧或者右侧小范围内的岸边即可，使用语义分割算法进行逐像素分类浪费了大量的计算资源，对于清洁小船而言，降低计算复杂度与推理时间更为重要，在此我们将车道线检测算法 Ultra Fast Structure-aware Deep Lane Detection[5]

加以改进用于我们的水岸线检测中。

### 2.2.1 算法原理

将车道线检测定义为寻找车道线在图像中某些行的位置的集合，即基于行方向上的位置选择、分类(row-based classification)，如图2.9所示，假设要检测一条车道线的图像大小为  $H \times W$ ，对于分割问题，我们需要处理  $H \times W$  个分类问题。由于方案是行向选择，假设在  $h$  个行上做选择，只需要处理  $h$  个行上的分类问题，只不过每行上的分类问题是  $W$  维的。因此这样就把原来  $H \times W$  个分类问题简化为了只需要  $h$  个分类问题，而且由于在哪些行上进行定位是可以人为设定的，因此  $h$  的大小可以按需设置，但一般  $h$  都是远小于图像高度  $H$  的。

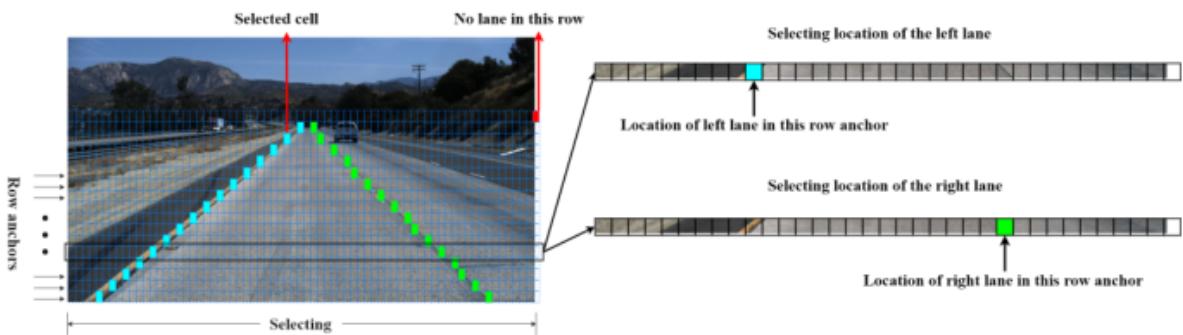


图 2.9: 基于行选择的车道线检测示意图

算法整体结构图如图2.10所示，Main branch 用于训练和测试，Auxiliary branch 只用于训练，在 Auxiliary branch 使用分割的交叉熵损失函数  $L_{seg}(2.2)$ 。考虑到连续性问题，每一行预测的位置也抛弃了传统的  $Loc_{i,j} = \arg \max_k P_{i,j,k}, k \in [1, w]$ ，而采用加权和的形式：

$$\begin{aligned} Prob_{i,j,:} &= \text{softmax}(P_{i,j,1:w}) \\ Loc_{i,j} &= \sum_{k=1}^w k \cdot Prob_{i,j,k} \end{aligned} \quad (2.10)$$

在主分支使用三个损失函数(车道线数量为  $C$ ，预测的概率为  $P \in \mathbb{R}^{C \times h \times (w+1)}$ ，one-hot 编码的真值为  $T \in \mathbb{R}^{C \times h \times (w+1)}$ ， $(w+1)$  主要是考虑到当前行可能没有车道线的情况)：

- 分类损失：对于选择的每一行，都是一个分类问题，同样使用交叉熵损失函数：

$$L_{cls} = \sum_{i=1}^C \sum_{j=1}^h L_{CE}(P_{i,j,:}, T_{i,j,:}) \quad (2.11)$$

- 连续性：考虑到车道线是连续的，因此使用一阶差分来定义连续性损失：

$$L_{sim} = \sum_{i=1}^C \sum_{j=1}^{h-1} \|P_{i,j,:} - P_{i,j+1,:}\|_1 \quad (2.12)$$

- 线性：考虑到大部分车道线是直的，因此采用二阶差分来定义线性损失：

$$L_{shp} = \sum_{i=1}^C \sum_{j=1}^{h-2} \| (Loc_{i,j,:} - Loc_{i,j+1,:}) - (Loc_{i,j+1,:} - Loc_{i,j+2,:}) \|_1 \quad (2.13)$$

最后总的损失通过三个损失函数加权得到，由于我们水岸线检测中不一定是直线，因此我们直接将二阶差分损失函数设置为 0，总损失计算如下：

$$L_{total} = L_{cls} + \alpha L_{sim} + 0 \cdot L_{shp} + \beta L_{seg} \quad (2.14)$$

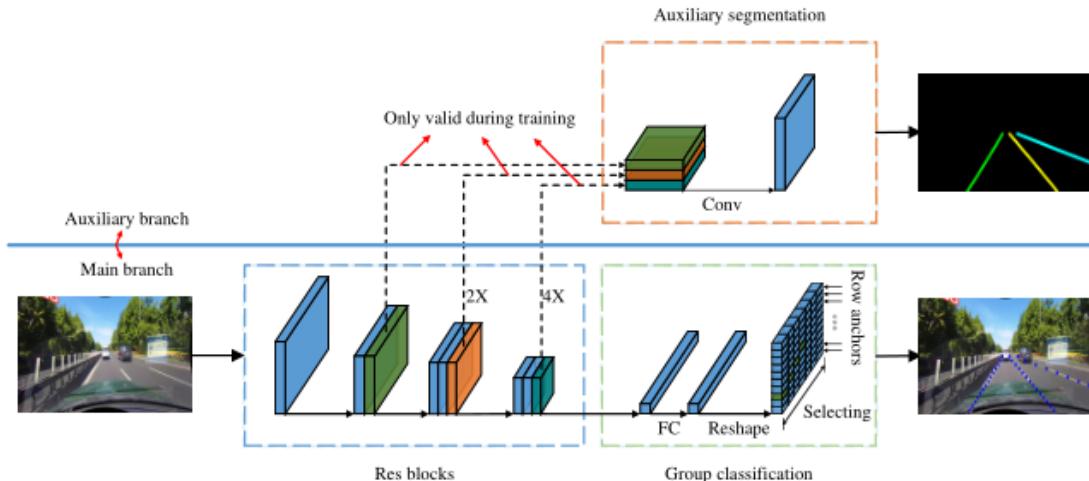


图 2.10: lane detection 算法结构图

## 2.2.2 数据标注

由于此种算法与分割算法使用的数据集不一样，因此我们首先使用 labelme[17] 标注了一批图像用于训练和测试，训练集共 2424 张图片，测试集共 607 张图片，训练集示例图像如图2.11所示。

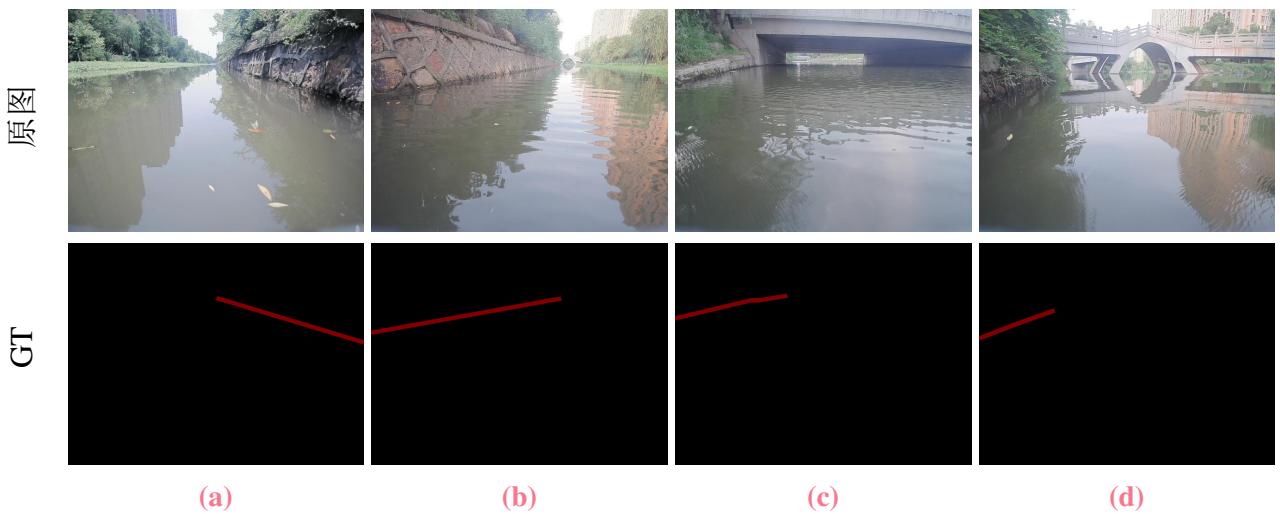


图 2.11: lane detection 数据示例图，第一行为原图，第二行为水岸线标记图 (GT)

### 2.2.3 实验结果

此部分实验使用 Nvidia RTX 1070 显卡进行实验，使用 pytorch 1.1.0+cuda 9.1 环境进行实验，实验采用自己标注的数据作为训练集和测试集，可视化结果如图2.12所示，可以看出使用此算法能够很精确的检测到比较常规的岸边，但是对于特殊场景（例如：桥下-图2.12(d)），检测有一定的误差。

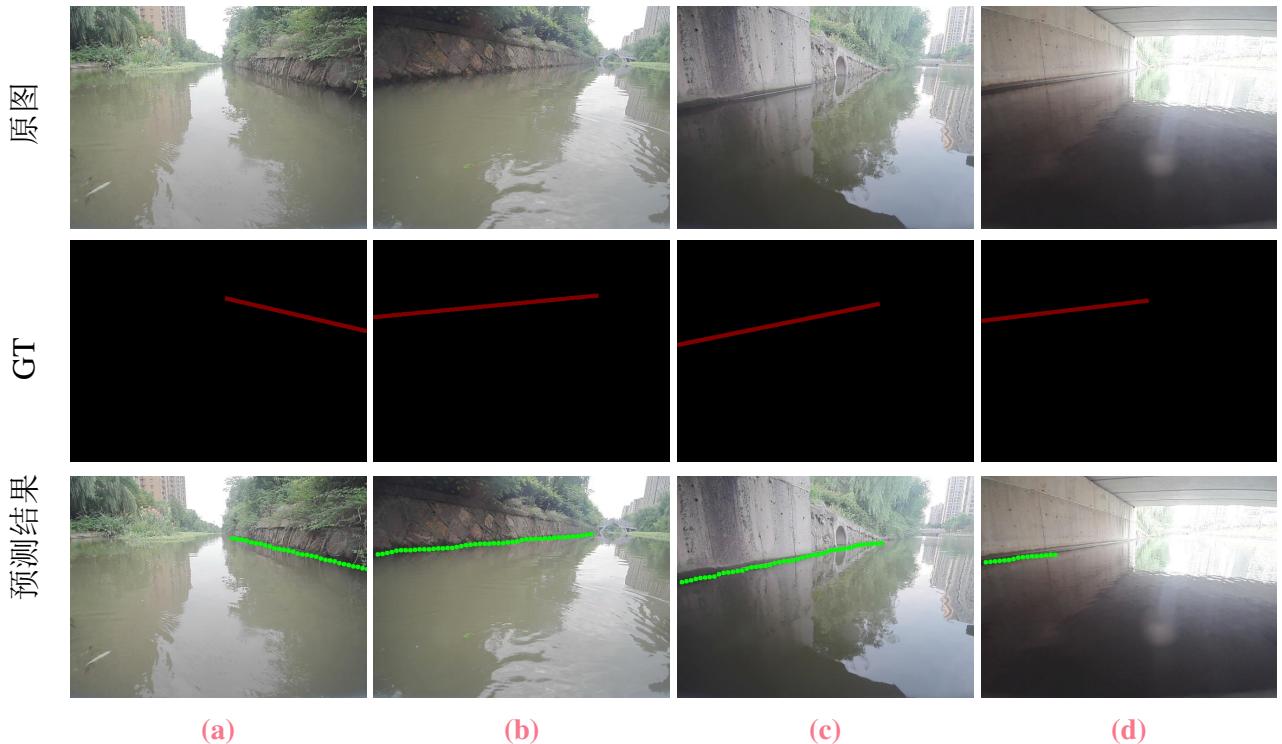


图 2.12: lane detection 算法实验可视化结果图

并且我们对比了三种基于语义分割的算法和此算法平均推理一张图所需要的时间如表2.2所示，可以看出 lane detection 的平均一张图的推理时间比其他三种基于语义分割的算法快 10 倍左右，在实际应用中可以更好的发挥作用。

**表 2.2:** Deeplabv3plus, SETR, Segformer 三种基于语义分割的算法和 lane detection 算法平均推理一张图片所用时间对比

算法	deeplabv3plus	SETR	Segformer	Lane Detection
时间 (s)	0.03342	0.04954	0.06455	<b>0.00337</b>

 **总结** 在实际应用中使用 lane detection 的算法更快，但是其鲁棒性不太好，可能是由于数据量较少的原因。针对鲁棒性和精度而言，Segformer 更好，可以利用 Segformer 算法去标记一部分数据，训练 lane detection 算法来获得更好的效果。

## 第三章 逆投影测距算法

在自动/辅助驾驶中，在前视摄像头拍摄的图像中，由于透视效应的存在，本来平行的事物，在图像中确实相交的。而逆投影 (Inverse Perspective Mapping, IPM) 变换就是消除这种透视效应，所以也叫逆透视。IPM 将摄像头拍摄的图像进行转换成鸟瞰图，本质上是二维平面到二维平面的转换。通过图像上的一个点，可以估计其真实位置。本章主要考虑两种基于逆投影的测距算法：

### 内容提要

□ Adaptive IPM

□ 基于单应性矩阵的测距

### 3.1 Adaptive IPM

Adaptive IPM[18] 主要是利用相机成像原理和空间立体几何关系来推导出照片上水面上的点对应的真实距离。对于图像像素坐标来说，坐标原点一般是左上角，而对于相机而言，图像的坐标原点在中心，如图3.1所示。

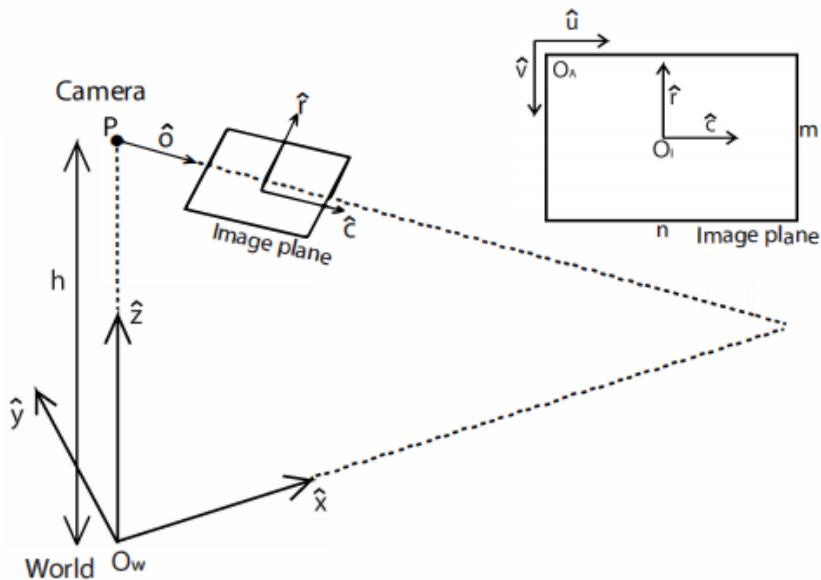


图 3.1: 相机坐标系、像素坐标系与世界坐标系的关系

根据几何关系，我们可以得到相机坐标系  $(\hat{r}, \hat{c})$  与像素坐标系  $(\hat{u}, \hat{v})$  的互化关系：

$$\begin{aligned} u(c) &= \frac{n+1}{2} + Kc \longleftrightarrow c(u) = \frac{1}{K} \left( u - \frac{n+1}{2} \right) \\ v(r) &= \frac{m+1}{2} - Kr \longleftrightarrow r(v) = \frac{1}{K} \left( \frac{m+1}{2} - v \right) \end{aligned} \quad (3.1)$$

其中  $\mathbf{K}$  是像素和米的变换比例 (px/m)。然后绘制出相机投影的侧视图如图3.2所示，可以推导出图像上点对应实际位置与相机的垂直距离  $\mathbf{X}$ (从示意图中可以看出，这个距离只和图像的垂直像素距离  $v$  有关):

$$\mathbf{X}(v) = \mathbf{h} \cot(\theta_0 - \theta(v)) \quad (3.2)$$

$$\theta(v) = \arctan\left(\frac{\mathbf{r}(v)}{\mathbf{f}_r}\right) \quad (3.3)$$

$$\tan(\alpha_r) = \frac{\mathbf{r}_{top}}{\mathbf{f}_r} \quad (3.4)$$

$$\mathbf{r}_{top} = \mathbf{r}(v=1) = \frac{1}{\mathbf{K}} \frac{\mathbf{m}-1}{2} \quad (3.5)$$

$$\mathbf{f}_r = \mathbf{r}_{top} \cot(\alpha_r) = \frac{\mathbf{m}-1}{2\mathbf{K}} \cot(\alpha_r) \quad (3.6)$$

因此有:

$$\begin{aligned} \theta(v) &= \arctan\left(\frac{\mathbf{r}(v)}{\mathbf{f}_r}\right) \\ &= \arctan\left(\left(1 - 2\frac{v-1}{\mathbf{m}-1}\right) \tan(\alpha_r)\right) \end{aligned} \quad (3.7)$$

$$\begin{aligned} \mathbf{X}(v) &= \mathbf{h} \cot(\theta_0 - \theta(v)) \\ &= \mathbf{h} \frac{\tan(\theta_0) \tan(\theta(v)) + 1}{\tan(\theta_0) - \tan(\theta(v))} \\ &= \mathbf{h} \frac{\tan(\theta_0) \left(1 - 2\frac{v-1}{\mathbf{m}-1}\right) \tan(\alpha_r) + 1}{\tan(\theta_0) - \left(1 - 2\frac{v-1}{\mathbf{m}-1}\right) \tan(\alpha_r)} \end{aligned} \quad (3.8)$$

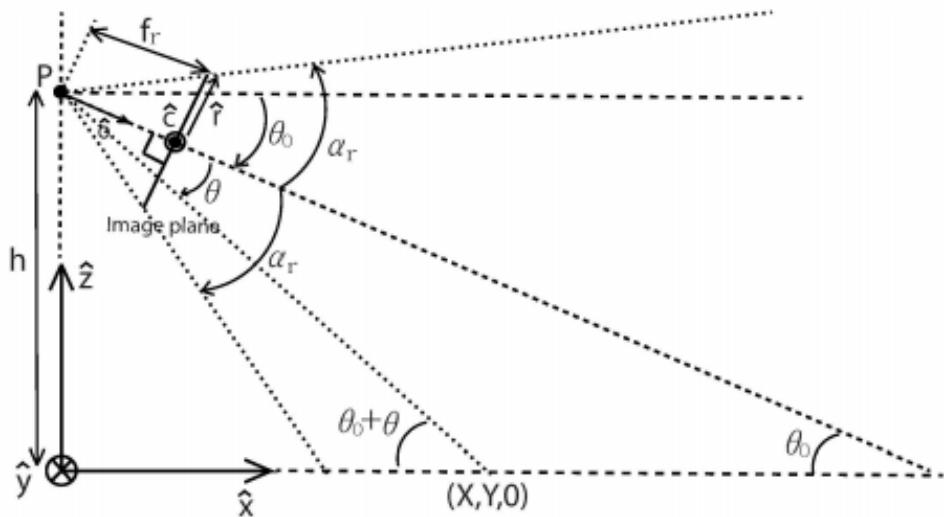


图 3.2: 相机投影的侧视图

得到  $\mathbf{X}(v)$  之后，再看俯视图3.3，可以看出图像上点对应实际位置与相机的水平距离  $\mathbf{Y}$  与垂直像素距离  $v$  和水平像素距离  $u$  均有关系，根据相似三角形的性质有：

$$\frac{\mathbf{Y}(u, v)}{\mathbf{X}(v)} = -\frac{\mathbf{c}}{\mathbf{f}_c} \quad (3.9)$$

根据几何性质有如下关系：

$$\tan(\alpha_c) = \frac{c_{right}}{f_c} \quad (3.10)$$

$$c_{right} = c(u = n) = \frac{1}{K} \left( \frac{n-1}{2} \right) \quad (3.11)$$

$$f_c = \frac{c_{right}}{\tan(\alpha_c)} = \frac{n-1}{2K \tan(\alpha_c)} \quad (3.12)$$

因此可以得到：

$$\begin{aligned} Y(u, v) &= -\frac{1}{K} \left( u - \frac{n+1}{2} \right) \frac{2K \tan(\alpha_c)}{n-1} X(v) \\ &= \left( 1 - 2 \frac{u-1}{n-1} \right) \tan(\alpha_c) X(v) \end{aligned} \quad (3.13)$$

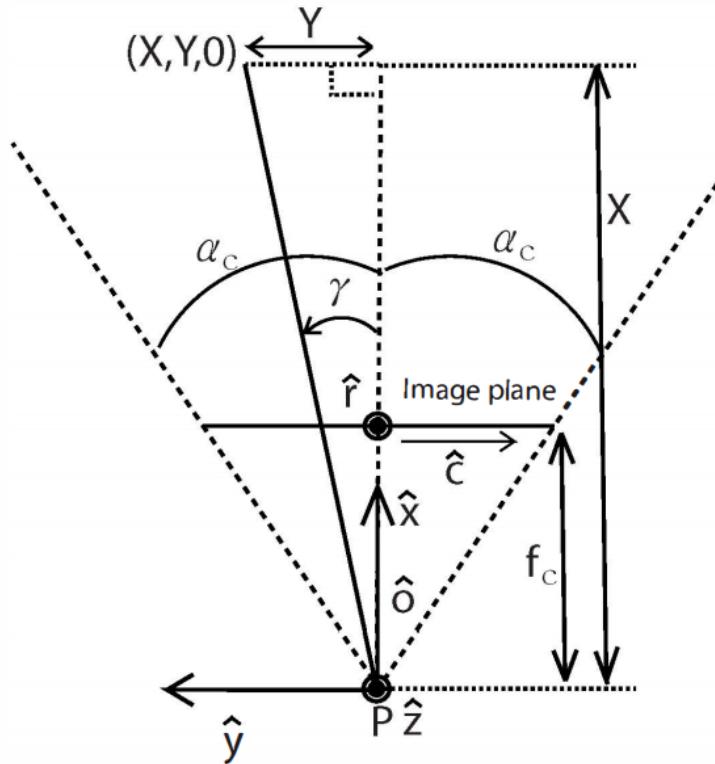


图 3.3: 相机投影的俯视图

这样便得到了静止状态下的图像上的点对应水面上实际位置距离摄像头的距离。但是当船行驶起来的时候，俯仰角会发生变化，即  $\theta_0$  会发生变化。如图3.4所示，此时相当于原先的  $\theta_0$  变成了  $\theta_0 + \theta_p$ ，因此我们可以得到动态变化下的距离公式：

$$\begin{aligned} X(v, \theta_p) &= h \frac{\tan(\theta_0 + \theta_p) (1 - 2 \frac{v-1}{m-1}) \tan(\alpha_r) + 1}{\tan(\theta_0 + \theta_p) - (1 - 2 \frac{v-1}{m-1}) \tan(\alpha_r)} \\ Y(u, v, \theta_p) &= \left( 1 - 2 \frac{u-1}{n-1} \right) \tan(\alpha_c) X(v, \theta_p) \end{aligned} \quad (3.14)$$

从示意图中可以看出，此场景只能计算出摄像头所在高度水平线以下的部分，而对于摄

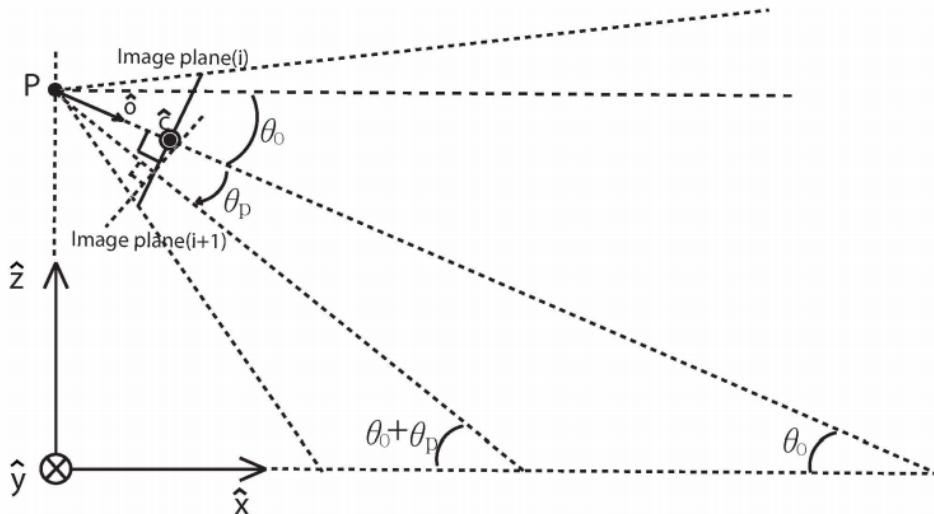


图 3.4: 俯仰角变化的相机投影的侧视图

像头平视、仰视的场景无法很好的处理，因此在小船上不太适用，在实际测试中也证实了这一点，3m 以内范围的误差大概有 1m 左右，非常大，因此最终没有选择此方案。

## 3.2 基于单应性矩阵的测距

在计算机视觉中，平面的单应性被定义为一个平面到另外一个平面的投影映射。因此一个二维平面上的点映射到摄像机 CCD 上的映射就是平面单应性的例子。如果相机 cmos 平面上的点到某一平面的映射使用齐次坐标，这种映射可以用矩阵相乘的方式表示，这个矩阵就是单应性矩阵。

$$s_i \begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} \sim H \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \quad (3.15)$$

其中单应性矩阵为  $H(3 \times 3$  矩阵)，由于单应性矩阵能进行归一化， $H_{33} = 1$ ， $x_i, y_i$  为图像上的点位置， $x'_i, y'_i$  为在逆投影平面的坐标， $s_i$  为齐次矩阵系数。

**求解单应性矩阵** 单应性矩阵除了  $H_{33} = 1$ ，还有有八个未知数，求解线性方程组至少需要四组对照点，每组对照点有  $x$  和  $y$  两个方程，联立所有方程求解。但实际求解单应性矩阵往往会获取多组数据求取迭代解，获得满足所有数据的最小误差解。opencv 提供了函数接口 cv2.findHmoography，通过 RANSAC 随机一致性采样处理个别误差较大的数据，最终通过 SVD 分解计算反投影误差 (back-projection error) 最小的一组迭代解。如下为反投影误差 (back-projection error) 表达式：

$$\sum_i \left( x'_i - \frac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2 + \left( y'_i - \frac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2 \quad (3.16)$$

**室内标定单应性矩阵** 将相机置于和船上位置相同高度下，将标定板放置在地面上，通过角点检测获取图像上的点，对于实际上的角点位置可以通过尺子测量得到。由于标定板呈矩形分

布，只需要测量四个角落的位置和标定板方格的尺寸既可获取整张图所有角点的位置。为降低测量误差，可以采集多张图像，例如不同尺寸的标定板不同位置的数据，一起求解。标定板如图3.5所示。



图 3.5: 单应性矩阵室内标定与测距测试

 **俯仰角校准** 在得到单应性矩阵后，可以对地面上的目标进行位置估计，如图3.5中的可乐瓶子（红点位置）在左前方（104cm,182cm）位置左右，逆投影估计的结果误差为 4cm，估计误差很低。但当相机发生姿态变换时，尤其时俯仰角的变化，会使得逆投影估计位置的精度瞬间增大，为了降低该误差，需要利用俯仰角补偿单应性矩阵。

此过程可以看作是先将摄像头拍摄的图像仅通过旋转操作转化为无俯仰角的情况，然后再通过标定好的单应性矩阵进行逆投影。本质上还是先通过一个仅旋转的单应性矩阵逆投影，再通过标定的单应性矩阵来逆投影。仅旋转的单应性矩阵可以很简单的计算如下：

$$H_p = K R K^{-1} \quad (3.17)$$

其中  $K$  为相机内参矩阵， $R$  为 3 维绕轴旋转矩阵，对于我们小船摄像头，仅需考虑俯仰变换，则旋转矩阵可定义如下：

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & \sin \theta \\ 0 & -\sin \theta & \cos \theta \end{bmatrix} \quad (3.18)$$

因此最终的带俯仰角校准的单应性矩阵变换为：

$$X = H \cdot H_p \cdot x = H(K R K^{-1})x \quad (3.19)$$

经过俯仰角补偿之后测试了相机带俯仰角情况下的测距，如图3.6所示，其中图3.6a表示俯视场景下的测距，误差大约 3cm，图3.6b表示仰视情况下的测距，误差同样 4cm 左右，可以看出经过俯仰角校准之后的测距误差非常小。



(a) 俯视场景

(b) 仰视场景

图 3.6: 俯仰角校准之后的单应性矩阵测距



**实际场景的测试** 为验证逆投影估计位置的精度，我们获取了实际场景（西安外事学院鱼化湖），通过激光雷达获取真实距离，对比相机估计的结果，如图3.7所示，右边坐标系红点为根据相机上的检测的水岸分界线逆投影得到的位置，蓝点为激光雷达的真实距离。对比发现经过俯仰角弥补后，远处的点位置估计不会发生较大误差。

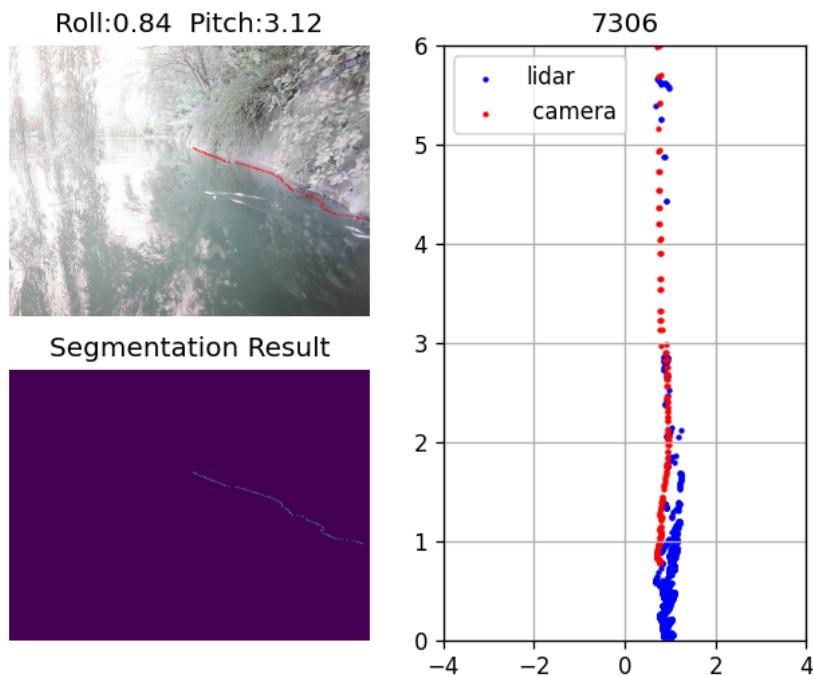


图 3.7: 单应性矩阵实际场景测试

**结论** 使用带俯仰角补偿的单应性矩阵逆投影法可以比较准确的测距，测距精度依赖于水岸线检测的精度，因此水岸线检测精度高时测距会比较好，可以满足贴边需求，并且单应性矩阵逆投影测距几乎不需要任何时间（只有简单的矩阵相乘），因此测距的速度也主要依赖于水岸线检测的速度，因此使用精度高且速度快的水岸线检测算法是一个非常重要的优化方向。

## 第四章 总结

在陕西欧卡电子智能科技有限公司博士生暑期实践期间主要负责无人驾驶清洁船贴边算法研究，主要包括两方面内容：水岸线检测和单目相机测距。

- **水岸线检测：**主要使用了基于图像语义分割的算法和基于车道线检测的算法，其中基于图像语义分割的 Segformer 的鲁棒性最好、效果最佳，但是推理时间较长，部署到 nano 平台会影响后续贴边速度；基于车道线检测的 ultra lane detection 算法速度最快，但是鲁棒性不太好，有一定的过拟合。因此可以利用 Segformer 去标记更多的数据，再去训练 ultra lane detection 来更进一步的优化。并且，在实践中完成了“Segformer 分割图像  $\Rightarrow$  labelme 格式 json 文件  $\Rightarrow$  输出可能的异常文件”的脚本，方便生成更多的数据集，此部分代码已上传到自己的 github： [https://github.com/luoyt14/orch\\_intern](https://github.com/luoyt14/orch_intern)；
- **单目相机测距：**主要完成了 Adaptive IPM 和单应性矩阵测距两种方法，Adaptive IPM 限制较多、误差较大，基本不能用于我们的水面测距场景；单应性矩阵测距法可以快速、低误差的完成水面测距，实际场景的测试实验也验证了单应性矩阵的可行性。

## 思想总结

纸上得来终觉浅，绝知此事要躬行。在短暂的陕西欧卡智能科技有限公司的暑期实践过程中，不仅将学校所学的东西运用到了实际的生产场景中，而且还学习了很多此前从未了解过的东西。

本次实践主要涉及两方面的工作，第一方面时利用计算机视觉的算法对无人清洁小船拍摄的场景进行水岸线的检测，此部分内容由于之前上过类似的课程，因此基本思路是有的，需要将目前最好的算法加以实现并用于无人清洁小船水岸线的检测。这部分内容带给我最大的收获是编程上的，将各种算法用于自己的场景，其中会出各种各样的 bug，将这些 bug 一一消除的过程使得我对于各种不同库的相似函数有了更深的了解。例如：pytorch 中图像的 shape 是 [Batchsize, Channels, Height, Width]，PIL.Image 库中图像的 size 为 [Width, Height]，cv2 中图像的 shape 是 [Height, Width, Channels]，这些图像 shape 的顺序都不一样，如果不搞清楚，在最后生成结果的时候就会出现很离谱的错误。而且还发现 Pillow 库 8.3.1 版本的一个 bug，`PIL.ImageOps.expand(image,border=0,fill=0)` 函数在图像 mode 为 ‘P’ 且 border 为 tuple 时会出错，目前已有人在官方 github 提了 issue，之后版本应该会修复。第二部分是利用单目相机进行测距，这部分的知识是之前从未学习过的，因此前期花了很久的时间去了解相机成像原理，查找文献来进行学习，在此非常感谢许浒大哥的指导，让我能够比较快速的学会这部分的内容，并且很好的完成了此部分的工作。

除了每日在公司写代码，偶尔也亲身去湖边操作小船，一方面可以采集新的数据来优化算法，另一方面亲手操作小船也非常快乐，仿佛拥有了一个遥控大玩具，体验到了小时候玩遥控小汽车的快乐。

在生活方面也有了很大的改变，作息规律，早上 9 点上班，下午 6 点下班，中午休息 2 小时，与学校自由的作息不同，规律作息也减少了熬夜的次数，算是一个很大的改变吧。公司偶尔也会有团建，大家一块去逛逛大唐不夜城，感受西安这个城市的魅力以及与北京那种快节奏生活的不同。

实践的时光是短暂的，也是快乐的，在顺利完成预定工作的同时，收获了很多学习上和生活上的知识。通过自己的努力为公司带来一点小小的收益，我认为就是有价值的，这可能也是学校博士生实践的根本目的吧，通过亲身的工作来为社会创造价值，我认为非常有意义。

最后感谢学校的组织联系以及欧卡公司给的实践机会，感谢欧卡程宇威、王培栋、许浒的指导，感谢小伙伴们——陈誉博、张广滨、赵春程的陪伴与帮助。

## 参考文献

- [1] 欧卡智舶. 陕西欧卡电子智能科技有限公司官方网站[Z]. <https://orca-tech.cn/>.
- [2] Liang-Chieh Chen and Yukun Zhu and George Papandreou and Florian Schroff and Hartwig Adam. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation[Z]. 2018. arXiv: [1802.02611 \[cs.CV\]](https://arxiv.org/abs/1802.02611).
- [3] Zheng, Sixiao and Lu, Jiachen and Zhao, Hengshuang and Zhu, Xiatian and Luo, Zekun and Wang, Yabiao and Fu, Yanwei and Feng, Jianfeng and Xiang, Tao and Torr, Philip HS and others. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. [S.l. : s.n.], 2021: 6881-6890.
- [4] Xie, Enze and Wang, Wenhai and Yu, Zhiding and Anandkumar, Anima and Alvarez, Jose M and Luo, Ping. SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers[J]. ArXiv preprint arXiv:2105.15203, 2021.
- [5] Qin, Zequn and Wang, Huanyu and Li, Xi. Ultra Fast Structure-aware Deep Lane Detection[C] //The European Conference on Computer Vision (ECCV). [S.l. : s.n.], 2020.
- [6] Alexey Dosovitskiy and Lucas Beyer and Alexander Kolesnikov and Dirk Weissenborn and Xiaohua Zhai and Thomas Unterthiner and Mostafa Dehghani and Matthias Minderer and Georg Heigold and Sylvain Gelly and Jakob Uszkoreit and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale[Z]. 2021. arXiv: [2010.11929 \[cs.CV\]](https://arxiv.org/abs/2010.11929).
- [7] Chen, Liang-Chieh and Papandreou, George and Schroff, Florian and Adam, Hartwig. Rethinking atrous convolution for semantic image segmentation[J]. ArXiv preprint arXiv:1706.05587, 2017.
- [8] Vaswani, Ashish and Shazeer, Noam and Parmar, Niki and Uszkoreit, Jakob and Jones, Llion and Gomez, Aidan N and Kaiser, ukasz and Polosukhin, Illia. Attention is all you need[C]// Advances in neural information processing systems. [S.l. : s.n.], 2017: 5998-6008.
- [9] Hugo Touvron and Matthieu Cord and Matthijs Douze and Francisco Massa and Alexandre Sablayrolles and Hervé Jégou. Training data-efficient image transformers & distillation through attention[J]. ArXiv preprint arXiv:2012.12877, 2020.
- [10] Zhao, Hengshuang and Shi, Jianping and Qi, Xiaojuan and Wang, Xiaogang and Jia, Jiaya. Pyramid scene parsing network[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. [S.l. : s.n.], 2017: 2881-2890.

- 
- [11] Wang, Wenhai and Xie, Enze and Li, Xiang and Fan, Deng-Ping and Song, Kaitao and Liang, Ding and Lu, Tong and Luo, Ping and Shao, Ling. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions[J]. ArXiv preprint arXiv:2102.12122, 2021.
  - [12] Zhou, Bolei and Zhao, Hang and Puig, Xavier and Fidler, Sanja and Barriuso, Adela and Torralba, Antonio. Scene parsing through ade20k dataset[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. [S.l. : s.n.], 2017: 633-641.
  - [13] He, Kaiming and Zhang, Xiangyu and Ren, Shaoqing and Sun, Jian. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. [S.l. : s.n.], 2016: 770-778.
  - [14] MMCV Contributors. MMCV: OpenMMLab Computer Vision Foundation[Z]. <https://github.com/open-mmlab/mmcv>. 2018.
  - [15] MM Segmentation Contributors. MM Segmentation: OpenMMLab Semantic Segmentation Toolbox and Benchmark[Z]. <https://github.com/open-mmlab/mmsegmentation>. 2020.
  - [16] Y. Cheng and M. Jiang and J. Zhu and Y. Liu. Are We Ready for Unmanned Surface Vehicles in Inland Waterways? The USVI inland Multisensor Dataset and Benchmark[J]. IEEE Robotics and Automation Letters, 2021, 6(2): 3964-3970. DOI: [10.1109/LRA.2021.3067271](https://doi.org/10.1109/LRA.2021.3067271).
  - [17] Kentaro Wada. Labelme: Image Polygonal Annotation with Python[Z]. <https://github.com/wkentaro/labelme>. 2016.
  - [18] Jeong, Jinyong and Kim, Ayoung. Adaptive inverse perspective mapping for lane map generation with SLAM[C]//2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI). [S.l. : s.n.], 2016: 38-41.