郑州大学
创新创业基础与工程设计实践项目

VisionVoyage-基于鱼眼相机与其他感知技术的自动驾驶仿真系统
详细设计报告

公司名称： IngenuityDrive-创智行科技有限公司

小组编号： 21级计算机类09组

团队成员： 徐梓航 郭顺 徐梦蝶 郑辰乐 陈自豪

赵柏茗 郭晓卿 蔡从轩 华勇 李景尧

指导老师： 程楠

所属学院： 计算机与人工智能学院

编订日期： 2024年3月

郑州大学 ZHENGZHOU UNIVERSITY

目录

1. 引言

本阶段是在需求分析，概要设计的基础上对项目进行详细设计。其中主要包括模块实现设计。

1.1 目的

目前大众对于自动驾驶的呼声越来越高，在这样的背景下，我们公司开发VisionVoyage软件是为了满足自动驾驶领域对于鱼眼相机和感知技术的需求，并推动自动驾驶技术的发展。随着自动驾驶技术的不断成熟和市场需求的增加，人们对于自动驾驶的期待越来越高，希望能够通过先进的技术实现更安全、高效的交通出行方式。

VisionVoyage软件的开发将有助于解决目前自动驾驶领域存在的一些挑战和问题，例如鱼眼数据集稀缺、鱼眼图像畸变处理、多摄像头组合的语义分割算法等方面的技术瓶颈。通过提供普通图像转鱼眼图像、仿真环境下各种传感器的图像获取与处理、虚拟驾驶体验和自动驾驶仿真等功能，我们的软件可以帮助研究人员和工程师更好地理解和应用鱼眼相机和其他传感器数据，从而提升自动驾驶系统的感知能力和安全性。

此外，由于当前市场上公开发布的鱼眼数据集相对较少，VisionVoyage的开发填补了这一空白，为研究人员和开发者提供了更多的实验和模拟环境，促进了自动驾驶领域的创新和进步。我们公司致力于通过VisionVoyage软件的推出，为自动驾驶技术的发展做出贡献，并满足大众对于更先进、更安全交通技术的迫切需求。

本文档以书面的形式准确的描述了软件需求，使开发者深入理解软件需求，为下一步的开发阶段提供坚实的文档支持。

1.2 参考资料

表 **1.2.1** 参考资料

文件名称	发表日期	出版单位	来源	备注
VisionVoyage需求规格说明书	/	/	创智行科技有限公司	
VisionVoyage概要设计说明书	/	/	创智行科技有限公司	
Autopep8官方文档	/	PyPI官网		参考

2. 开发环境

表 2.1.1 参考资料

工具名称	作用	备注
VSCode	编程	Code Editor

PyCharm	编程	IDE
XML	作为配置文件	eXtensible Markup Language
YAML	作为配置文件	YAML Ain’t a Markup Language
Shell	作为Linux OS脚本文件	.sh
Batchfile	作为Windows OS脚本文件	.bat
Commadfile	作为Mac OS脚本文件	.command
PySide6	模板框架	前端应用的框架
QSS	美化UI界面，实现界面和程序的分离，快速切换皮肤	Qt样式表
User Interface File	XML格式，包含了各种控件（如按钮、文本框、列表框等）及其布局和属性	.ui
Python	后端	编写简易脚本
C++	后端	重写部分重要Python脚本以提升性能、UE4编程
Makefile	自动构建和管理类UNIX OS（Mac OS、Windows OS）下脚本的工具	
CMake	跨平台C/C++构建工具，用于管理软件构建过程	

3. 模块逻辑结构

3.1 模块功能

①普通图像鱼眼化

我司计划设计一个投影变换算法来将普通图像转为鱼眼图像，一个立方体贴图算法将前后左右上五个视角的图像拼接成一个鱼眼图像，供用户大致了解鱼眼图像和普通真空相机图像的区别，为VisionVoyage后续功能的使用铺路。

②拍摄鱼眼数据集

由于当前市场上公开发布的鱼眼数据集相对较少，VisionVoyage的开发填补了这一空白，为研究人员和开发者提供了更多的实验和模拟环境，用户可以根据自己的需求选择需要拍摄的鱼眼数据集类型。

③仿真传感器

VisionVoyage提供了RGB针孔相机、语义分割相机、实例分割相机、深度相机、激光雷达、事件相机、光流相机、鱼眼相机等众多自动驾驶感知技术所需传感器，供用户选择自己所需传感器来进行学习研究。

④驾驶仿真

VisionVoyage提供了仿真环境，用户可以选择用键盘控制汽车行驶或者自动驾驶，汽车压线或者碰撞时，VisionVoyage会根据激光雷达和相机返回的数据来判断并提醒用户，汽车时速、帧率、坐标等关键信息也会展示在窗口上，也提供天气转换、转换摄像头视角、更换车辆等功能。

自动驾驶功能是我司集成前后左右四个鱼眼相机和激光雷达等传感器返回的信息优化自动驾驶算法并部署到VisionVoyage仿真环境上，能有效避障，且遵循交通规则，为开发人员自己设计的自动驾驶算法提供了一个蓝图。

⑤上传

用户使用我司的拍摄鱼眼数据集功能或自己的数据集或车辆行驶视频，可以直接上传到我司VisionVoyage接口上，通过处理，最终呈现给用户的是一个分割好的图像并输出所包含的语义信息或视频流实时分割。

⑥我的

“我的”模块是软件中的个性化定制中心，为用户提供定制化服务和个性化体验。首先，用户可以根据自己的喜好选择适合自己的软件主题，实现暗色和亮色两种风格的切换。其次，用户可以通过“我的图像”功能查看经过软件图像转鱼眼操作的图片以及原始图片，带来全新的视觉体验。此外，用户还可以提交个性化需求，我司会使用UE4 Editor进行需求定制所需的仿真地图和车辆模型等UE资产，以获得独特的自动驾驶仿真体验。最后，用户可以通过“联系我们”功能方便快捷地与我们取得联系，提出需求、问题或意见，我们将及时回复并为用户提供所需的支持和帮助。

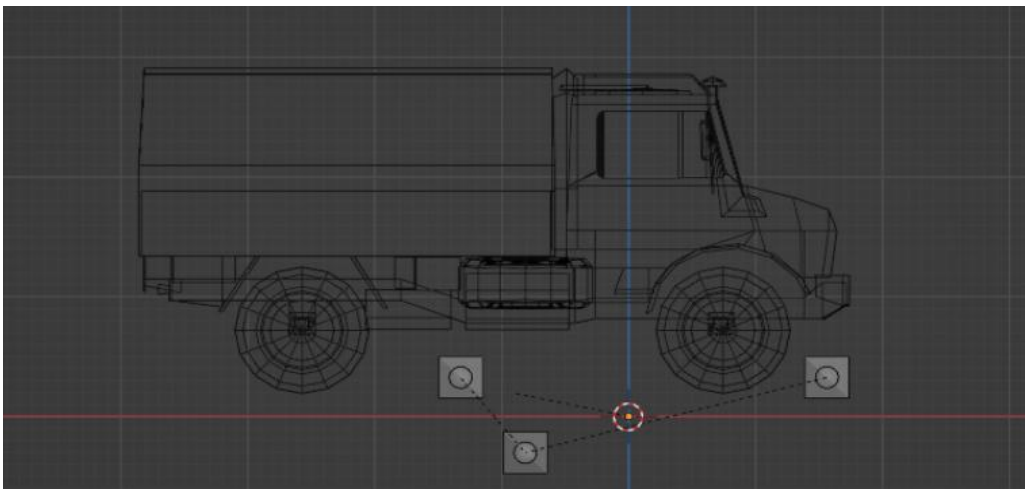

图3.1.1 定制车辆模型

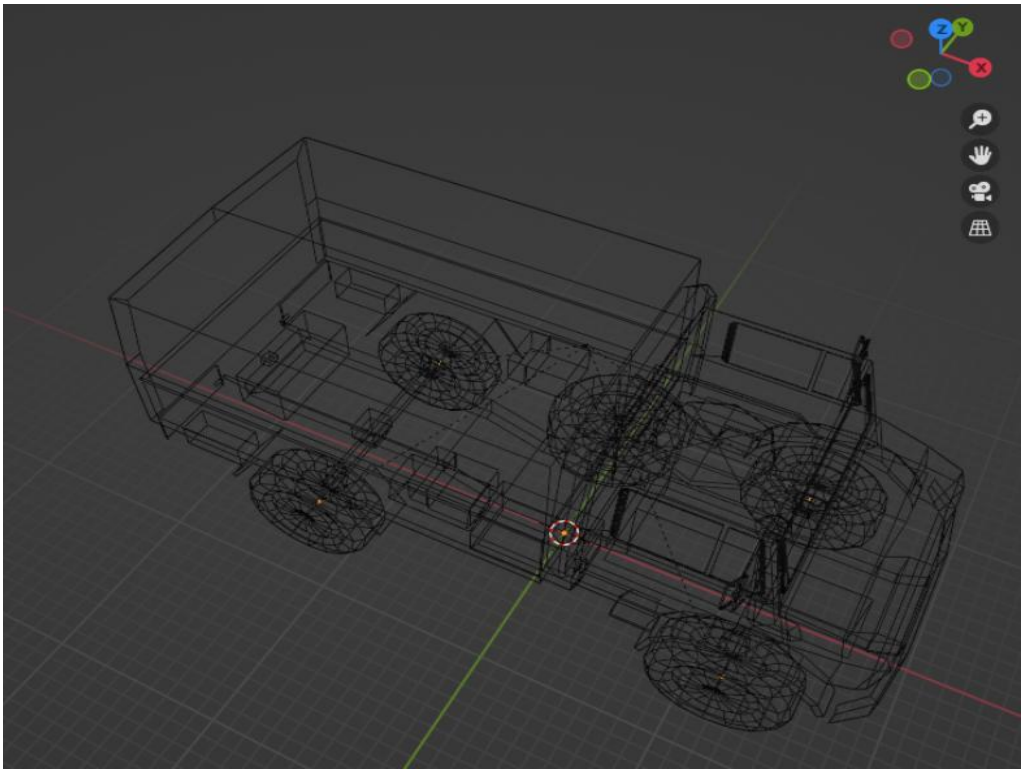

图3.1.2 定制车辆模型

3.2 模块组成

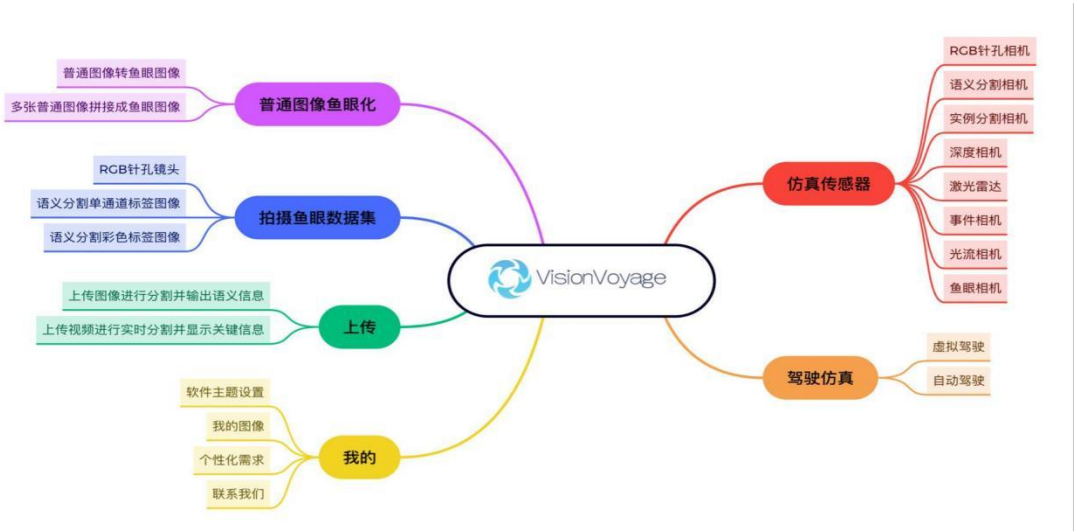

图 **3.2.1** 系统功能模块图

4. 模块实现设计

4.1 普通图像鱼眼化

（1）投影变换算法

①描述

我司计划设计一个投影变换算法来将普通图像转为鱼眼图像，一个立方体贴图算法将前后左右上五个视角的图像拼接成一个鱼眼图像，供用户大致了解鱼眼图像和普通真空相机图像的区别，为VisionVoyage后续功能的使用铺路。

②功能

该模块主要由两个功能模块组成，包括图像的输入与图像的处理。

图像的输入由用户通过GUI的FILE BOX选择本地图片获取。

图像的处理将标准图像中的点坐标转换为鱼眼图像中的点坐标。

③输入项

合法的本地图像。

④输出项

与输入图像相同格式的图像。

⑤算法

使用投影变换算法将透视图像上的像素重新映射到鱼眼图像上。面对鱼眼数据集缺乏的问题，可以利用已有的普通图像，将普通图像中的像素重映射到鱼眼图像上，由于普通图像的视场角相对较小，会导致生成的鱼眼图像的视场角进一步缩小。因为普通图像无法完全捕捉到鱼眼镜头的广角特性，所以在重映射过程中，部分细节和广阔的场景可能会受到限制，导致鱼眼图像的质量和准确性出现折损。如果应用场景中只需要获得大致的视野范围或者对细节要求不高，那么通过重映射可以提供适用的数据。

常见的语义分割数据集都是使用针孔相机拍摄的，其原理也就是我们所熟悉的小孔成像。针孔相机模型采用直线投影方式，简单来说，如果物体的线条原本是直的，那么按照直线投影在拍出来的照片里，它的线条也是直的，不会变弯。

直线投影的缺点就是视场角受限。在现实里，符合直线投影的无畸变的全画幅超广角镜头，焦距差不多只能做到10mm，视场角最广大概为130°，再往上会变得非常困难。可以从公式的角度来理解这一点。公式(1)为θ为主轴与入射光线的夹角，r为像点与主点的距离，f为焦距。

$$r = f\tan\theta (1)$$

因为tan90°是无穷大，也就是说半视场角越接近90°，像高会越来越大，直到变成无穷。因此在直线投影下，如果要做180°的广角镜头，那么你的物平面和像平面都需要无穷大，这是无法实现的。如果我们还想要再广的视野，那就要摆脱直线投影的约束，对视场角进行压缩。这就来到了鱼眼镜头的领域。

鱼眼镜头视角比普通的超广角镜头还要更广，常常可以超过180°。在大众摄影里，鱼眼镜头因为夸张的变形而不怎么受待见，但是它在工业、科研、安防、军事等领域却有着非常广泛的运用。因为机器对于符合人眼观感没有需求，它只是需要更大的视角。在针孔相机模型里，成像是从物平面到像平面。也就是直线投影是一种从平面到平面的投影。而在鱼眼相机模型里，通过引入桶形畸变，其成像是从物球面到像平面。也就是鱼眼投影是从球面到平面的投影。

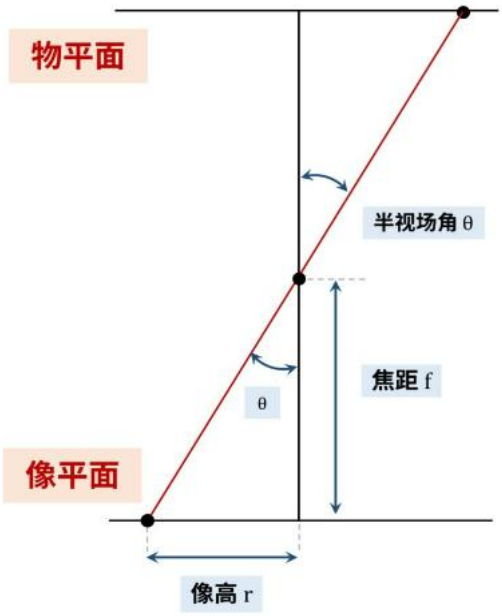

图4.1.1 针孔相机模型

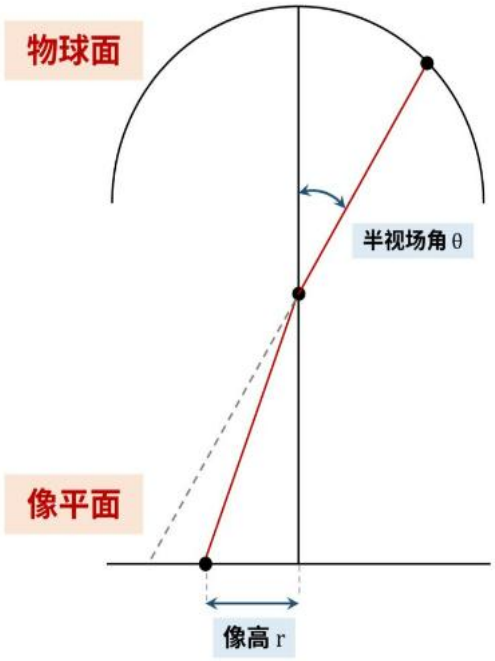

图4.1.2 鱼眼相机模型

鱼眼镜头有很多种投影模型，常见的鱼眼投影有等距投影、体视投影、等积投影，正交投影。

我们的鱼眼镜头均采用最常用的等矩投影。其公式如(2)所示，从图4.1.3可以看出当半视场角超过90°时，像高并不会无线大，所以制作出的鱼眼镜头的视场角可以超过180°。

$$r = f\theta \quad (2)$$

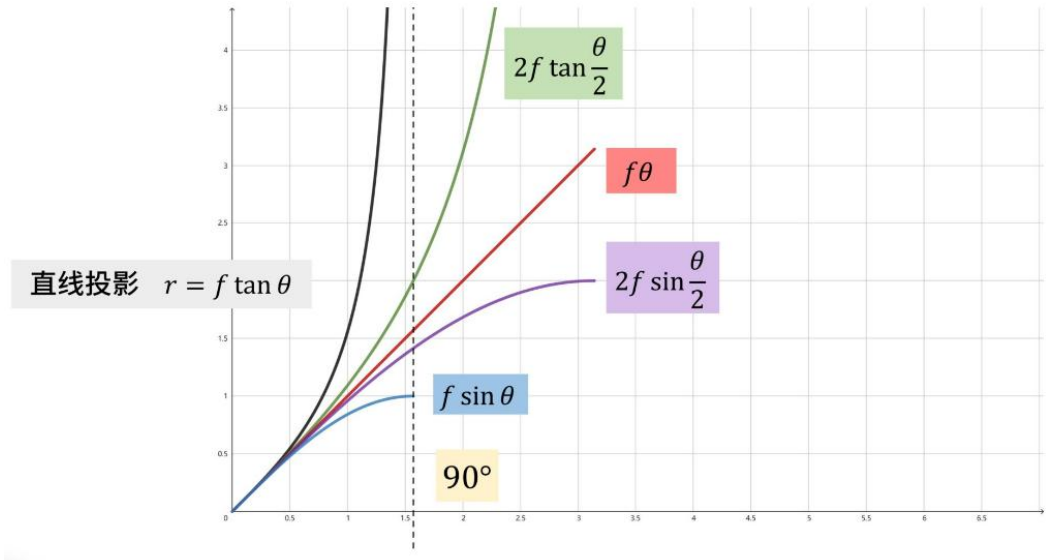

图4.1.3 直线投影与四种经典的鱼眼投影模型公式

我们的投影变换算法假设直线投影模型与鱼眼投影模型的焦距f，通过两者投影公式

之间的关系，将普通图像上的像素重新映射到了鱼眼图像上，通过改变焦距，还可以得到畸变程度不同的鱼眼图像，改变焦距的方式又分为随机变焦和固定变焦，效果图如图4.1.4所示。

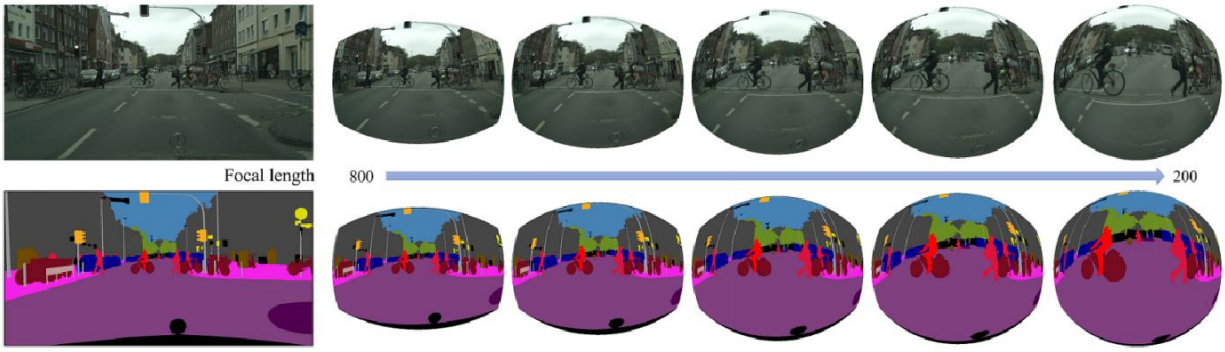

图4.1.4 图像变换结果。左边是原始的彩色图像和注释，右边是焦距从200到800的变换后的鱼眼图像和注释

⑥运行限制：

VisionVoyage详细设计说明书以需求规格说明书和概要设计为依据，设计的前提和限制条件是需求规格说明书规定的需求内容和概要设计描述的内容，对于没有规定和描述的内容，不纳入设计范畴。如需求发生变更，则首先履行需求变更手续，更改需求规格说明书，然后相应的更改概要设计内容，此后才更改相应的详细设计内容。

（2）立方体贴图算法

通过在UE4搭建的仿真三维场景中，为三维场景中的鱼眼相机标定参数，进一步模拟鱼眼相机按照标定参数在所述三维场景中按照相应位置和角度采集多个预定方向上的图片，并将这些图片渲染成一个立方体贴图，从而根据预定的鱼眼成像模型模拟光线的真实路线，来确定立方体贴图中的采样点与成像平面上的鱼眼成像位置之间的关系，进而完成鱼眼图像各个像素的颜色采样，并渲染成鱼眼图像。

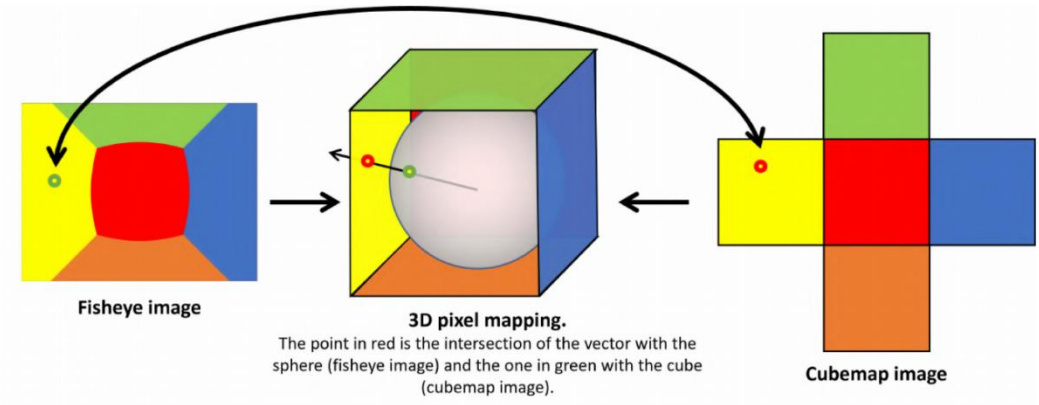

图4.1.5 立方体贴图

图4.1.6 鱼眼图像生成流程图

4.2 拍摄鱼眼数据集

①描述

VisionVoyage为研究人员和开发者提供了更多的实验和模拟环境，用户可以根据自己的需求选择需要拍摄的鱼眼数据集类型。

②功能

拍摄鱼眼数据集功能由 2 个模块实现，分别是：VisionVoyage Server，VisionVoyage Client。其中，VisionVoyage Server由UE4 Editor打包项目发布，VisionVoyage Client是用户实际使用的GUI客户端。

③输出项

鱼眼RGB数据集和语言语义分割RAW图像。

④VisionVoyage Server构建

分图层构建虚拟场景：

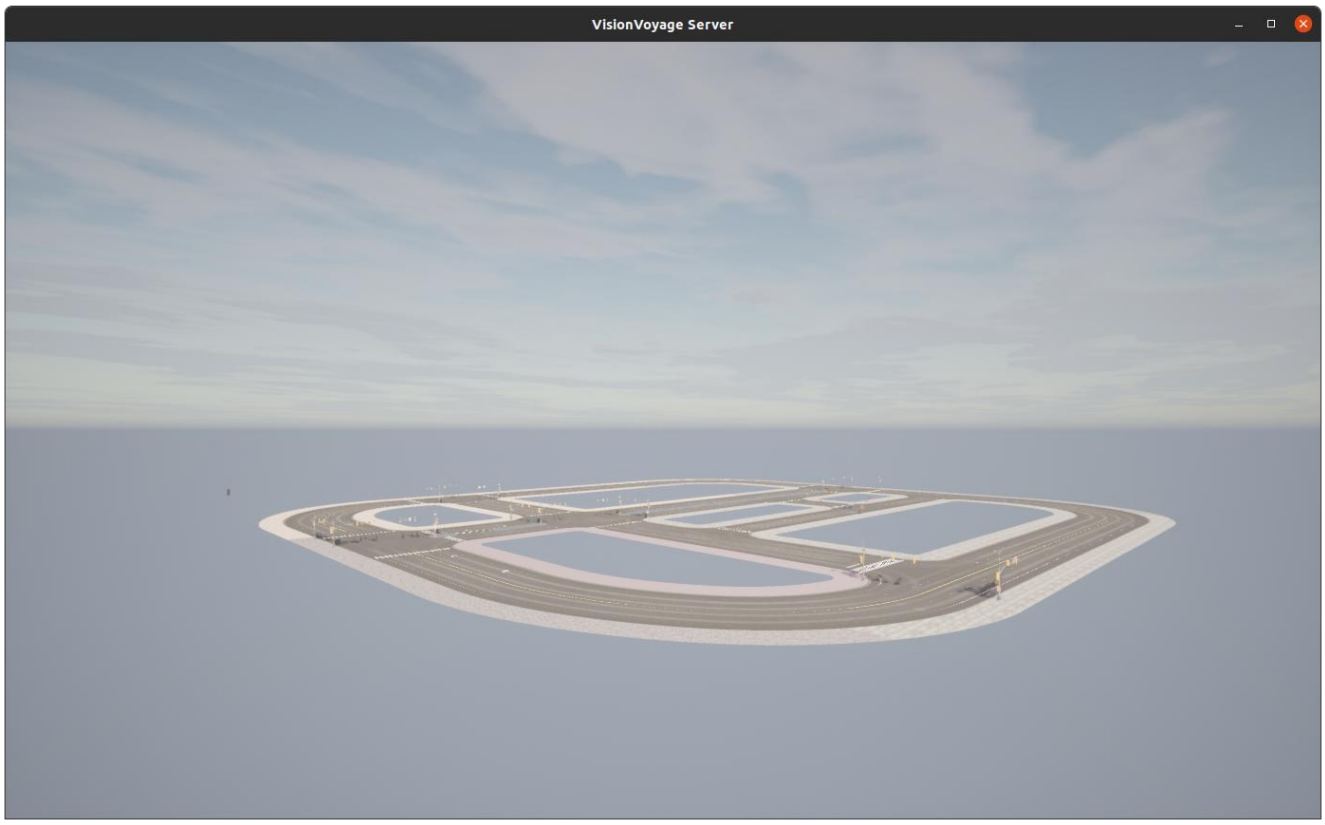

图4.2.1 NONE

图4.2.2 墙壁

图4.2.3 路灯

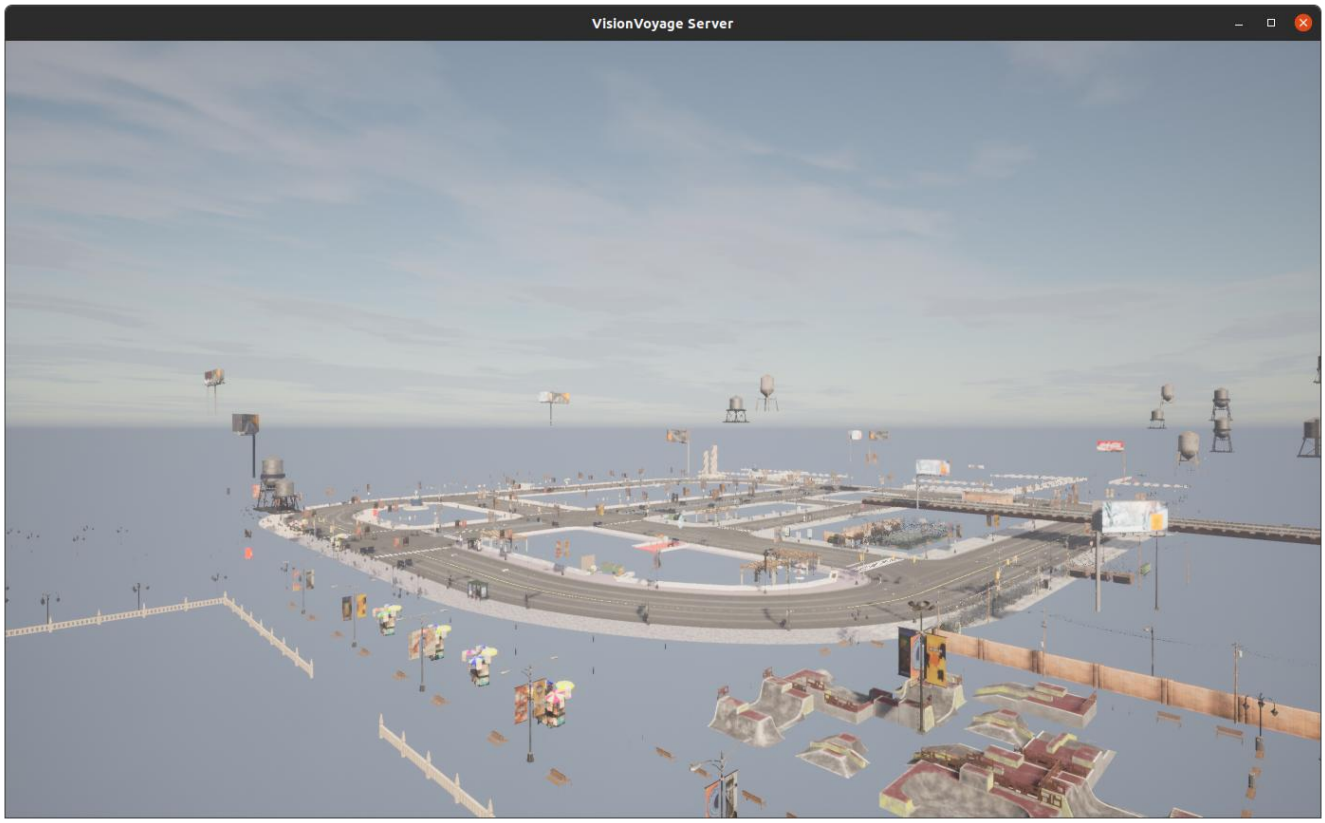

图4.2.4 道具

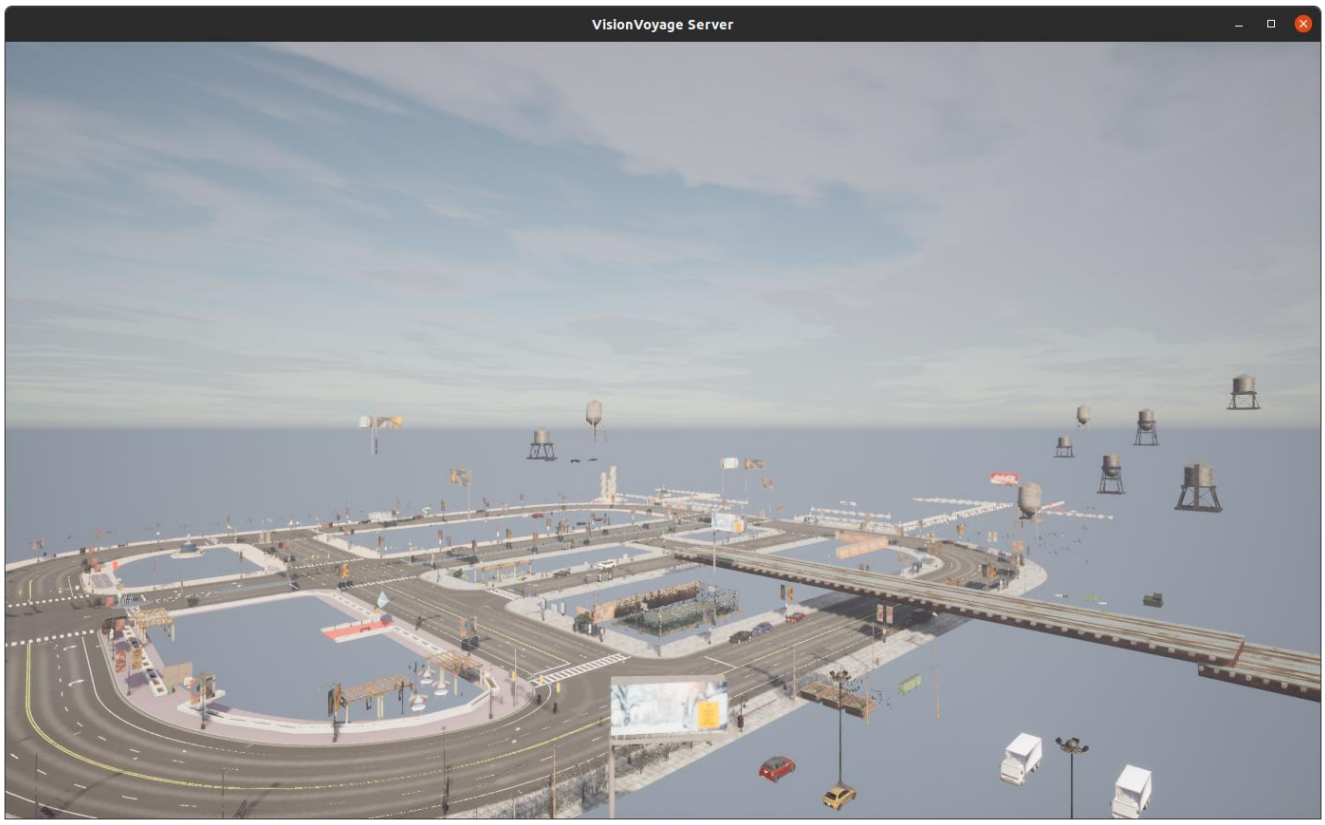

图4.2.5 停放车辆

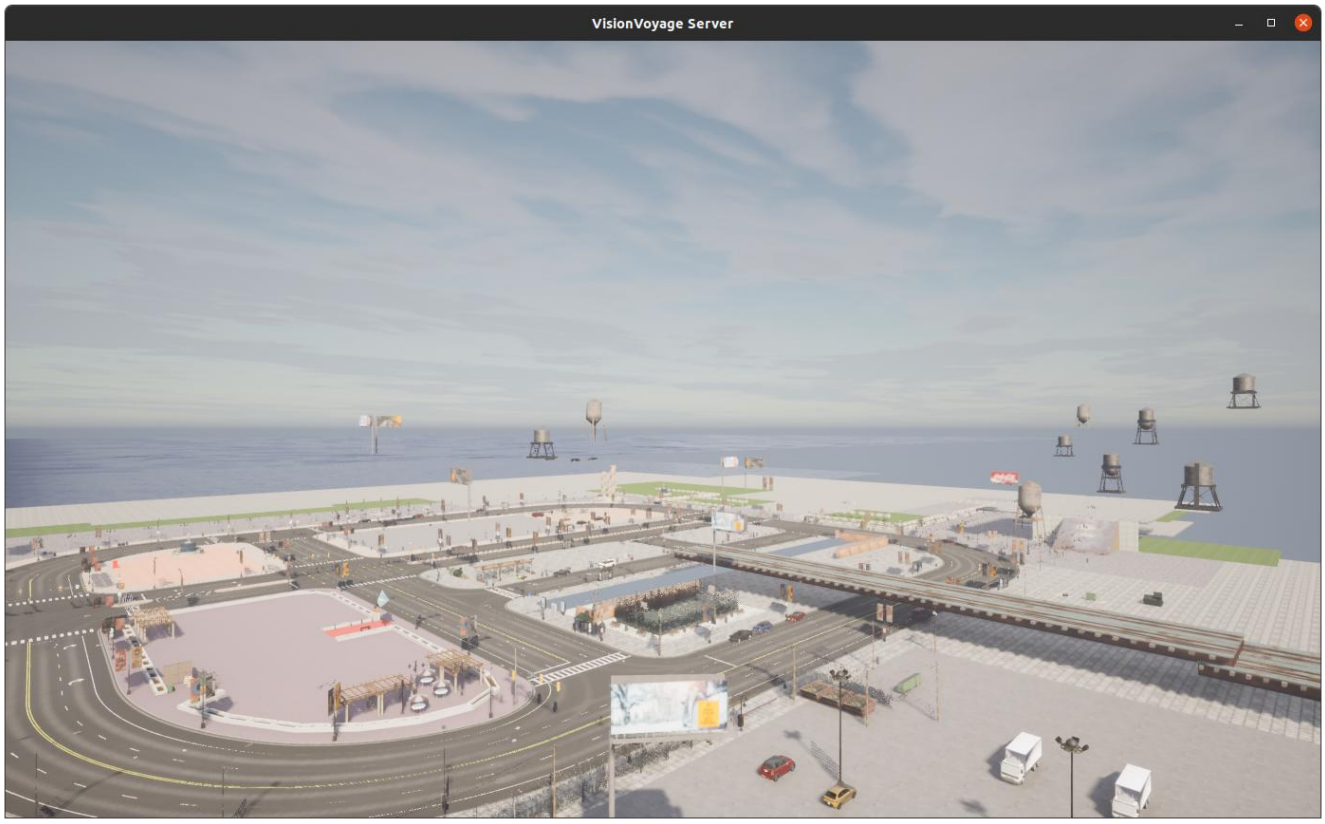

图4.2.6 地面

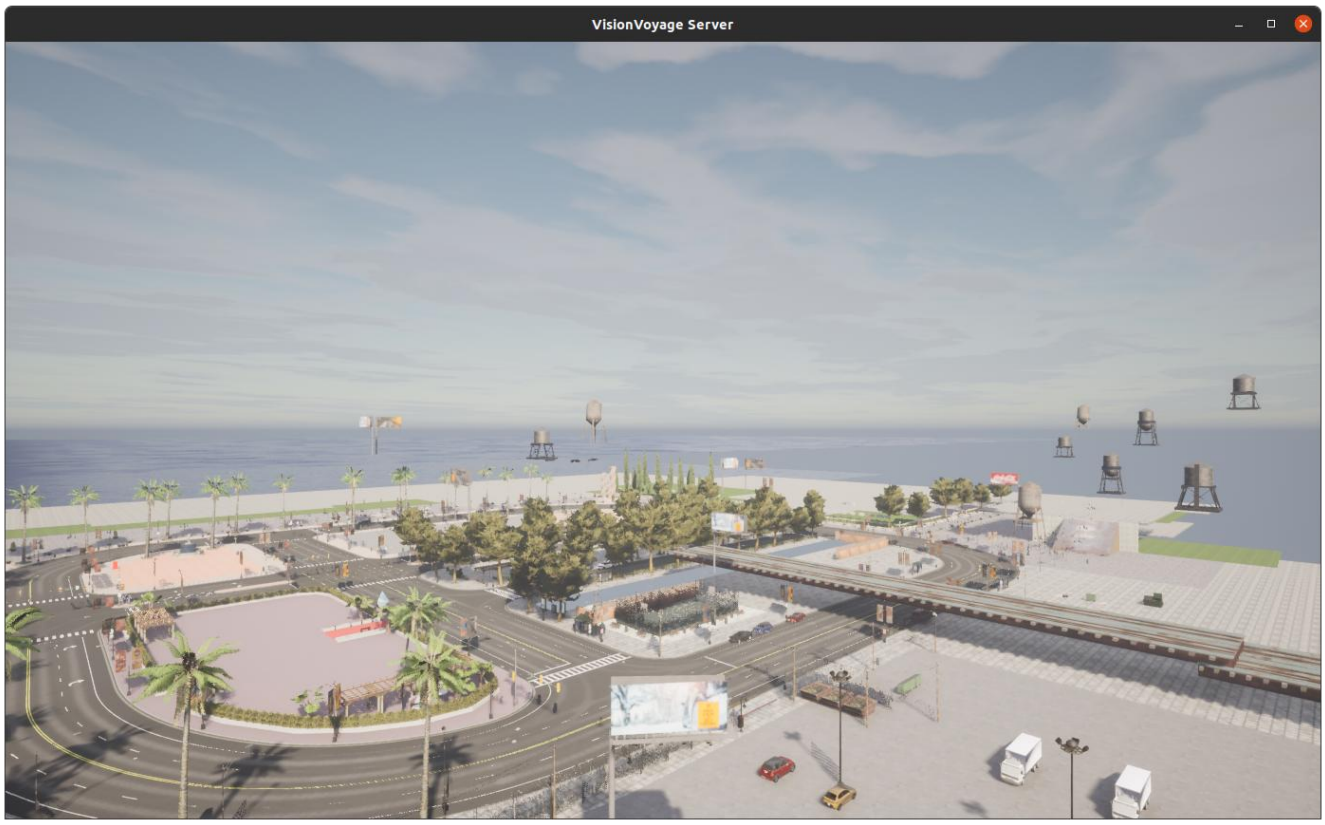

图4.2.7 植被

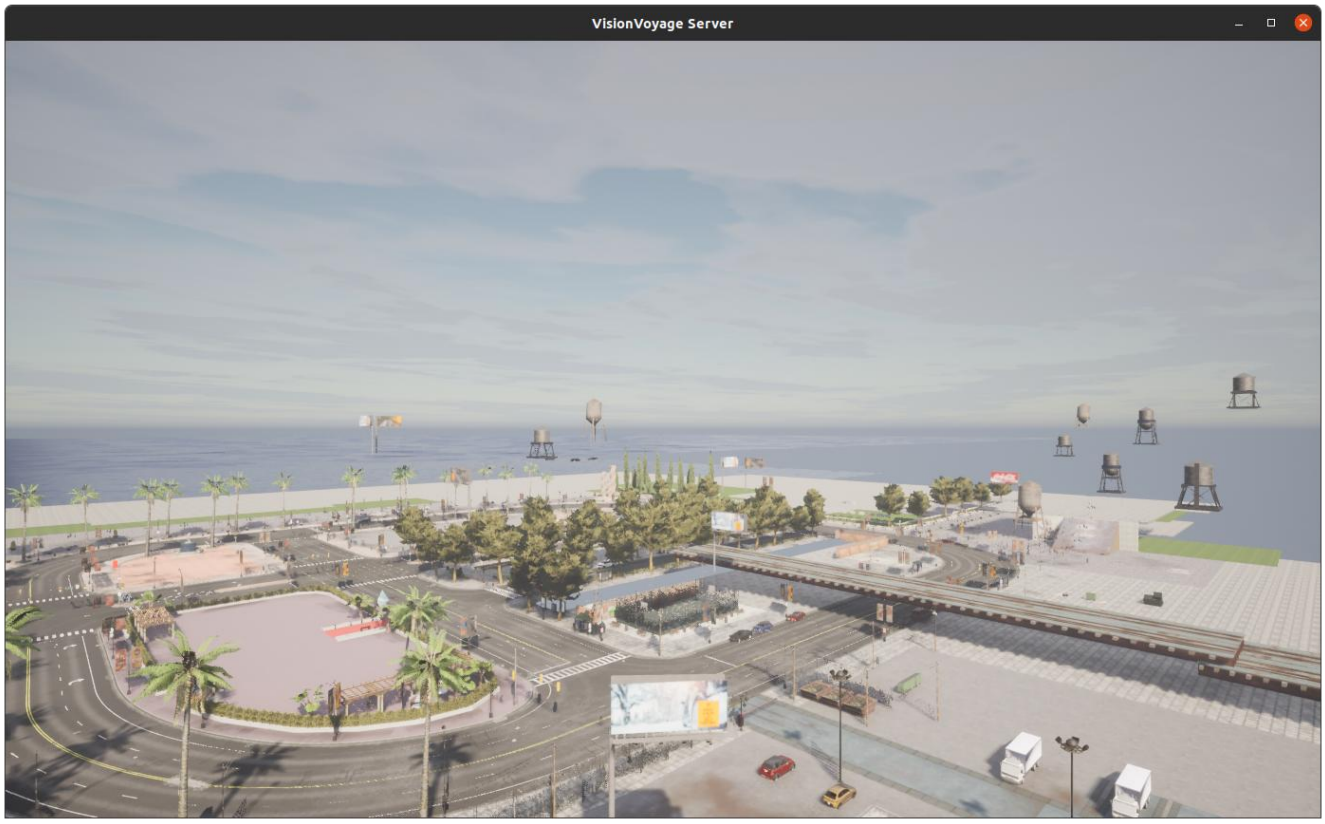

图4.2.8 贴图细节

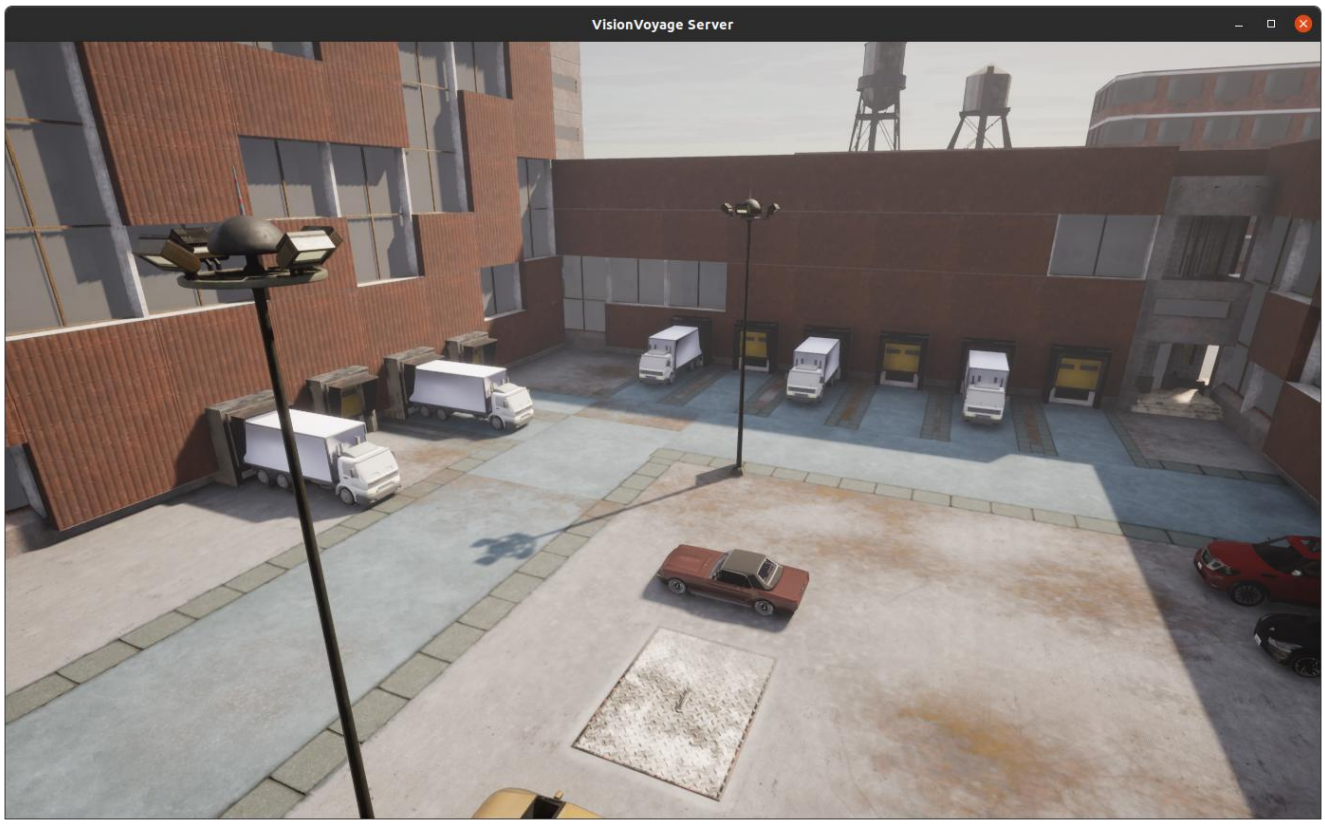

图4.2.9 建筑物

⑤UE4中鱼眼传感器构建（部分代码）

```
class USceneCaptureComponentCube; // UE 捕获场景组件
class UDrawFrustumComponent;      // UE 绘制视锥体组件
class UTextureRenderTargetCube;   // UE 立方体纹理渲染目标
class UStaticMeshComponent;       // UE 静态网格组件

UCLASS() // 基于 UE
class AFisheyeSensor : public ASensor
{
    GENERATED_BODY()

    friend class FReader;

public:
    AFisheyeSensor(const FObjectInitializer &ObjectInitializer);

    static FActorDefinition GetSensorDefinition();

    UFUNCTION(BlueprintCallable)
    float GetImageWidth() const;

    UFUNCTION(BlueprintCallable)
    float GetImageHeight() const;

    UFUNCTION(BlueprintCallable)
    float GetFOV() const;

    // 在渲染线程中发送像素
    template <typename TSensor>
```

```
29.     static void SendPixelsInRenderThread(TSensor &Sensor, float MaxAngle, float SizeX,
float SizeY, float Fx, float Fy, float Cx, float Cy, float D1, float D2, float D3, float
D4);
30.
31.     void Set(const FActorDescription &ActorDescription) override;
32.
33.     // 渲染目标纹理
34.     UPROPERTY(EditAnywhere)
35.     UTextureRenderTargetCube *CaptureRenderTarget = nullptr;
36.
37.     // 鱼眼传感器组件
38.     UPROPERTY(EditAnywhere)
39.     USceneCaptureComponentCube *Fisheye = nullptr;
40.
41.     UPROPERTY(EditAnywhere)
42.     float MaxAngle = 0;
43.
44.     UPROPERTY(EditAnywhere)
45.     float XSize = 0;
46.
47.     UPROPERTY(EditAnywhere)
48.     float YSize = 0;
49.
50.     UPROPERTY(EditAnywhere)
51.     float Fx = 0;
52.
53.     UPROPERTY(EditAnywhere)
54.     float Fy = 0;
55.
56.     UPROPERTY(EditAnywhere)
57.     float Cx = 0;
58.
59.     UPROPERTY(EditAnywhere)
60.     float Cy = 0;
61.
62.     UPROPERTY(EditAnywhere)
63.     float D1 = 0;
64.
65.     UPROPERTY(EditAnywhere)
66.     float D2 = 0;
67.
68.     UPROPERTY(EditAnywhere)
69.     float D3 = 0;
70.
71.     UPROPERTY(EditAnywhere)
72.     float D4 = 0;
73.
74. protected:
75.     virtual void BeginPlay() override;
76.
77.     // 物理更新前调用
78.     virtual void PrePhysTick(float DeltaTime) override;
79.
80.     // 物理更新后调用
81.     virtual void PostPhysTick(UWorld *World, ELevelTick TickType, float DeltaTime)
override;
82.
83.     virtual void EndPlay(const EEndPlayReason::Type EndPlayReason) override;
84.
85.     bool ReadyToCapture = false;
86. };
```

```
FActorDefinition AFisheyeSensor::GetSensorDefinition()
{
    auto Definition = UActorBlueprintFunctionLibrary::MakeGenericSensorDefinition(
        TEXT("camera"),
        TEXT("fisheye"));

    FActorVariation XSize;
    XSize.Id = TEXT("x_size");
    XSize.Type = EActorAttributeType::Float;
    XSize.RecommendedValues = {TEXT("1000.0")};
    XSize.bRestrictToRecommended = false;

    FActorVariation YSize;
    YSize.Id = TEXT("y_size");
    YSize.Type = EActorAttributeType::Float;
    YSize.RecommendedValues = {TEXT("900.0")};
    YSize.bRestrictToRecommended = false;

    FActorVariation MaxAngle;
    MaxAngle.Id = TEXT("max_angle");
    MaxAngle.Type = EActorAttributeType::Float;
    MaxAngle.RecommendedValues = {TEXT("200.0")};
    MaxAngle.bRestrictToRecommended = false;

    FActorVariation Fx;
    Fx.Id = TEXT("f_x");
    Fx.Type = EActorAttributeType::Float;
    Fx.RecommendedValues = {TEXT("300.0")};
    Fx.bRestrictToRecommended = false;

    FActorVariation Fy;
    Fy.Id = TEXT("f_y");
    Fy.Type = EActorAttributeType::Float;
    Fy.RecommendedValues = {TEXT("300.0")};
    Fy.bRestrictToRecommended = false;

    FActorVariation Cx;
    Cx.Id = TEXT("c_x");
    Cx.Type = EActorAttributeType::Float;
    Cx.RecommendedValues = {TEXT("600.0")};
    Cx.bRestrictToRecommended = false;

    FActorVariation Cy;
    Cy.Id = TEXT("c_y");
    Cy.Type = EActorAttributeType::Float;
    Cy.RecommendedValues = {TEXT("400.0")};
    Cy.bRestrictToRecommended = false;

    FActorVariation D1;
    D1.Id = TEXT("d_1");
    D1.Type = EActorAttributeType::Float;
    D1.RecommendedValues = {TEXT("0.0")};
    D1.bRestrictToRecommended = false;

    FActorVariation D2;
    D2.Id = TEXT("d_2");
    D2.Type = EActorAttributeType::Float;
    D2.RecommendedValues = {TEXT("0.0")};
    D2.bRestrictToRecommended = false;

    FActorVariation D3;
    D3.Id = TEXT("d_3");
    D3.Type = EActorAttributeType::Float;
```

```
64.     D3.RecommendedValues = {TEXT("0.0")};
65.     D3.bRestrictToRecommended = false;
66.
67.     FActorVariation D4;
68.     D4.Id = TEXT("d_4");
69.     D4.Type = EActorAttributeType::Float;
70.     D4.RecommendedValues = {TEXT("0.0")};
71.     D4.bRestrictToRecommended = false;
72.
73.     Definition.Variations.Append({XSize, YSize, MaxAngle, Fx, Fy, Cx, Cy, D1, D2, D3,
D4});
74.
75.     return Definition;
76. }
77.
78. void AFisheyeSensor::Set(const FActorDescription &Description)
79. {
80.     Super::Set(Description);
81.
82.     XSize = UActorBlueprintFunctionLibrary::RetrieveActorAttributeToFloat(
83.         "x_size",
84.         Description.Variations,
85.         1000.0f);
86.
87.     YSize = UActorBlueprintFunctionLibrary::RetrieveActorAttributeToFloat(
88.         "y_size",
89.         Description.Variations,
90.         900.0f);
91.
92.     MaxAngle = UActorBlueprintFunctionLibrary::RetrieveActorAttributeToFloat(
93.         "max_angle",
94.         Description.Variations,
95.         200.0f);
96.
97.     Fx = UActorBlueprintFunctionLibrary::RetrieveActorAttributeToFloat(
98.         "f_x",
99.         Description.Variations,
100.        300.0f);
101.
102.    Fy = UActorBlueprintFunctionLibrary::RetrieveActorAttributeToFloat(
103.        "f_y",
104.        Description.Variations,
105.        300.0f);
106.
107.    Cx = UActorBlueprintFunctionLibrary::RetrieveActorAttributeToFloat(
108.        "c_x",
109.        Description.Variations,
110.        600.0f);
111.
112.    Cy = UActorBlueprintFunctionLibrary::RetrieveActorAttributeToFloat(
113.        "c_y",
114.        Description.Variations,
115.        400.0f);
116.
117.    D1 = UActorBlueprintFunctionLibrary::RetrieveActorAttributeToFloat(
118.        "d_1",
119.        Description.Variations,
120.        0.0f);
121.
122.    D2 = UActorBlueprintFunctionLibrary::RetrieveActorAttributeToFloat(
123.        "d_2",
124.        Description.Variations,
125.        0.0f);
```

```
126.
127.     D3 = UActorBlueprintFunctionLibrary::RetrieveActorAttributeToFloat(
128.         "d_3",
129.         Description.Variations,
130.         0.0f);
131.
132.     D4 = UActorBlueprintFunctionLibrary::RetrieveActorAttributeToFloat(
133.         "d_4",
134.         Description.Variations,
135.         0.0f);
136.
137.     CaptureRenderTarget->SizeX = XSize;
138. }
```

之后通过UE Render Targets渲染到纹理上，之后通过自定义蓝图对渲染的纹理进行后处理实现各种效果。

⑥模型:

各种模型为开源UE4素材库。

4.3 仿真传感器

①描述

通过UE4的摄像头系统，构建多种传感器供用户使用。

②功能

提供了RGB针孔相机、语义分割相机、实例分割相机、深度相机、激光雷达、事件相机、光流相机、鱼眼相机等众多自动驾驶感知技术所需传感器，供用户选择自己所需传感器来进行学习研究。

③UE4摄像头系统

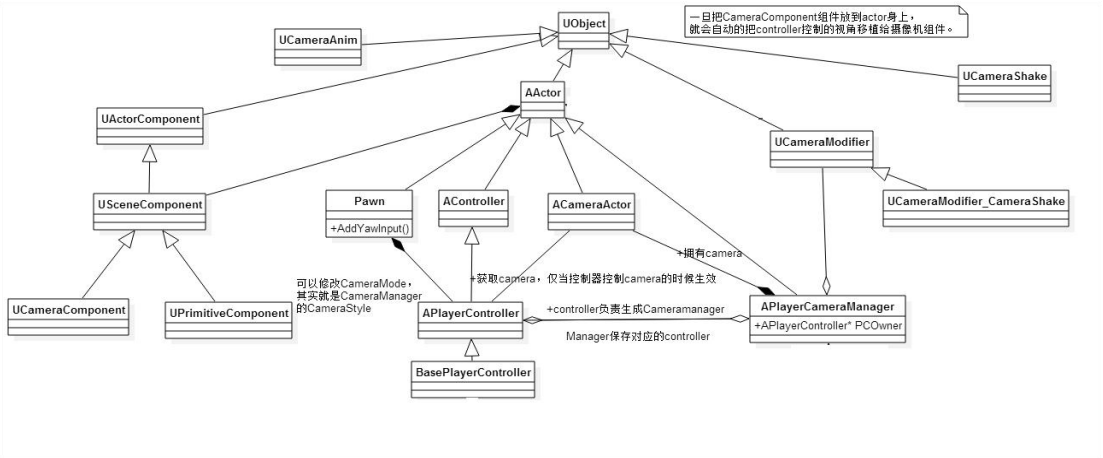

图4.3.1 UE4摄像头相关类图

图4.3.1 UE4 Editor摄像头属性

4.4 驾驶仿真

①描述

VisionVoyage提供了仿真环境，用户可以选择用键盘控制汽车行驶或者自动驾驶。

②功能

汽车压线或者碰撞时，VisionVoyage会提醒用户，汽车时速、帧率、坐标等关键信息也会展示在窗口上，也提供天气转换、转换摄像头视角等功能，目前还在开发中。

图4.3.1 测试画面

4.5 上传

①描述

用户使用我司的拍摄鱼眼数据集功能或自己的数据集或车辆行驶视频，可以直接上传到我司VisionVoyage接口上，通过处理，最终呈现给用户的是一个分割好的图像并输出所包含的语义信息或视频流实时分割。

②功能

通过我司的预训练权重对用户上传的图像或视频进行处理。

③输出项

带有语义信息的图像或视频。

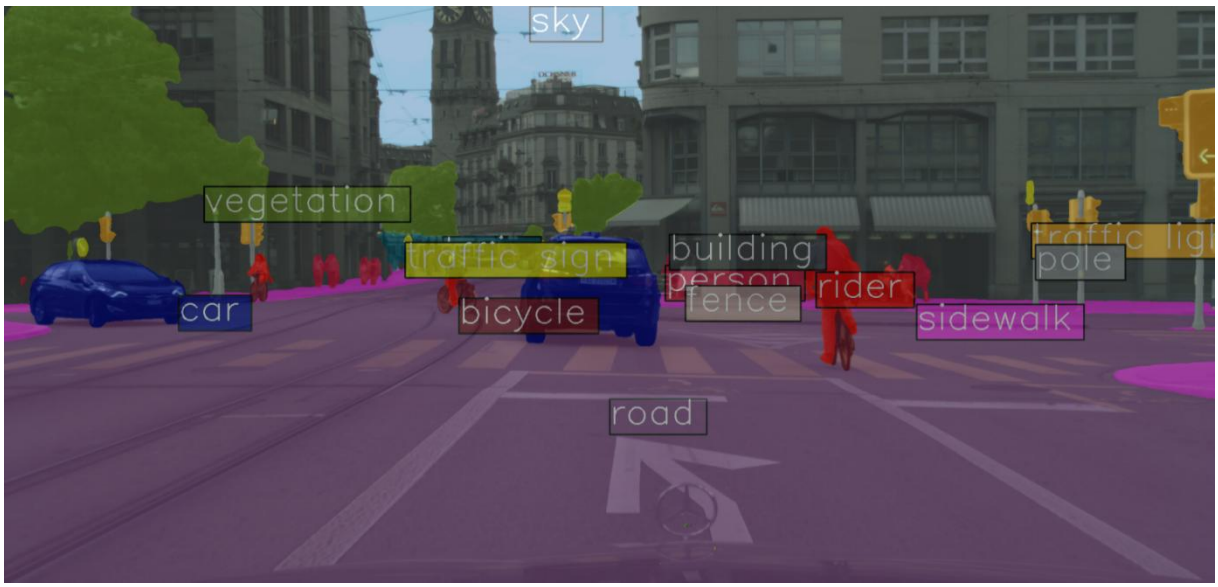

图4.5.1 普通图像分割示例图

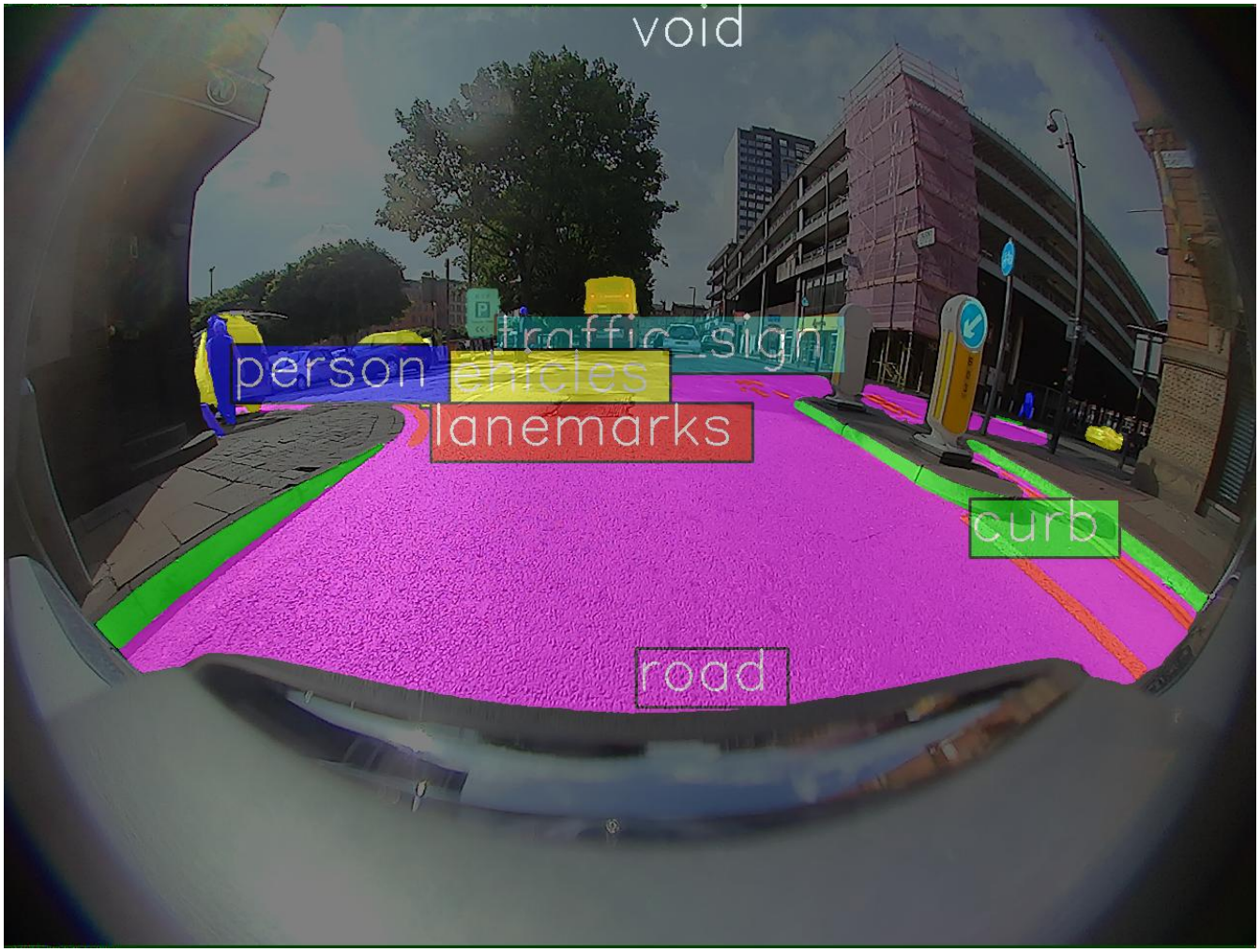

图4.5.2 鱼眼图像分割示例图

图4.5.3 视频分割示例图

4.6 我的

①描述

软件中的个性化定制中心，为用户提供定制化服务和个性化体验。

②功能

用户可以根据自己的喜好选择适合自己的软件主题，实现暗色和亮色两种风格的切换。其次，用户可以通过“我的图像”功能查看经过软件图像转鱼眼操作的图片以及原始图片，带来全新的视觉体验。此外，用户还可以提交个性化需求，我司会使用UE4 Editor进行需求定制所需的仿真地图和车辆模型等UE资产，以获得独特的自动驾驶仿真体验。最后，用户可以通过“联系我们”功能方便快捷地与我们取得联系，提出需求、问题或意见，我们将及时回复并为用户提供所需的支持和帮助。

③实现

通过QSS（Qt样式表）设计暗色、亮色两种风格主题，用qrc存储本地资源，再通过pyside6-designer综合设计GUI。查看图像功能分Windows OS、Linux OS、Mac OS三种系统通过调用不同系统API实现相同功能。

图4.6.1 暗色主题

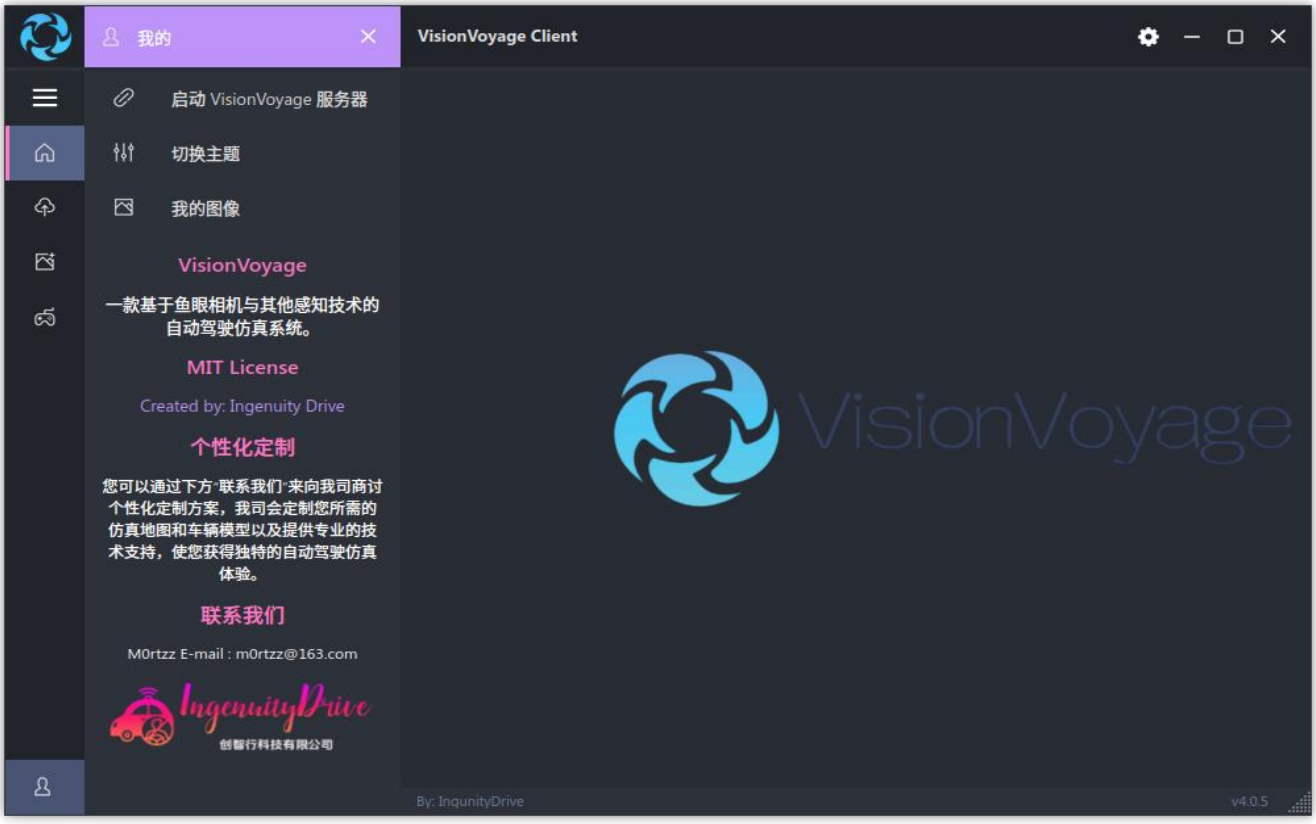

图4.6.2 “我的”模块侧边栏

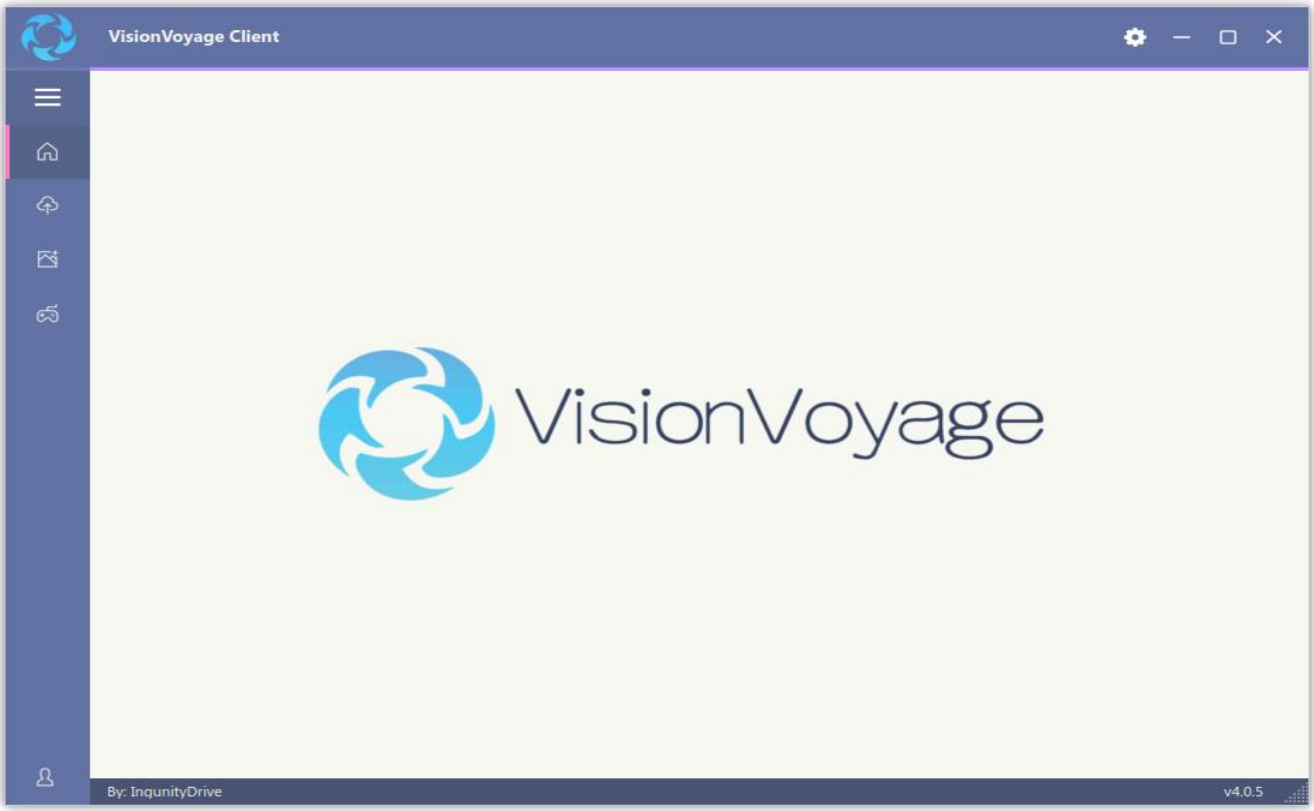

图4.6.2 亮色主题

5. 界面设计

页面设计的大致思路如下：

图5.1 主界面

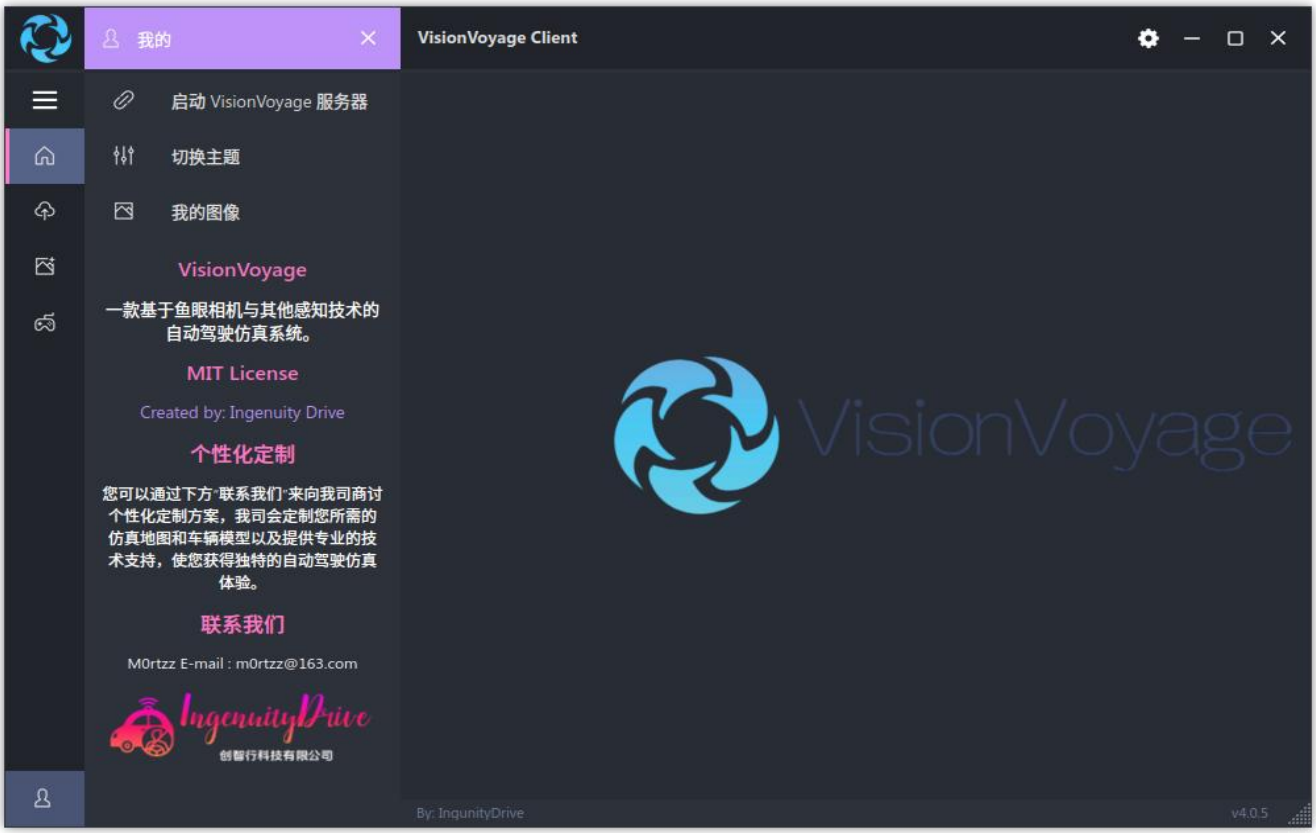

图5.2 “我的”模块

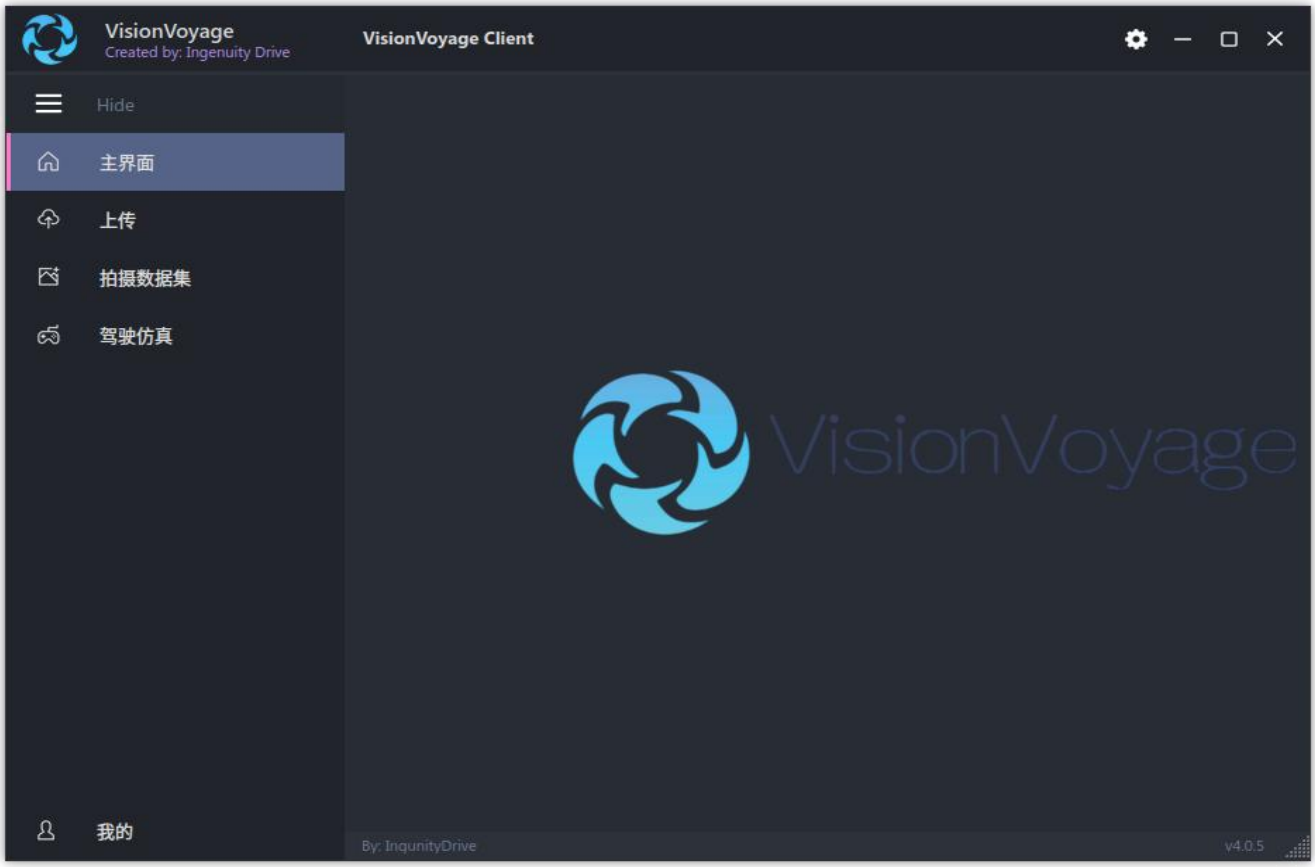

图5.3 侧边栏

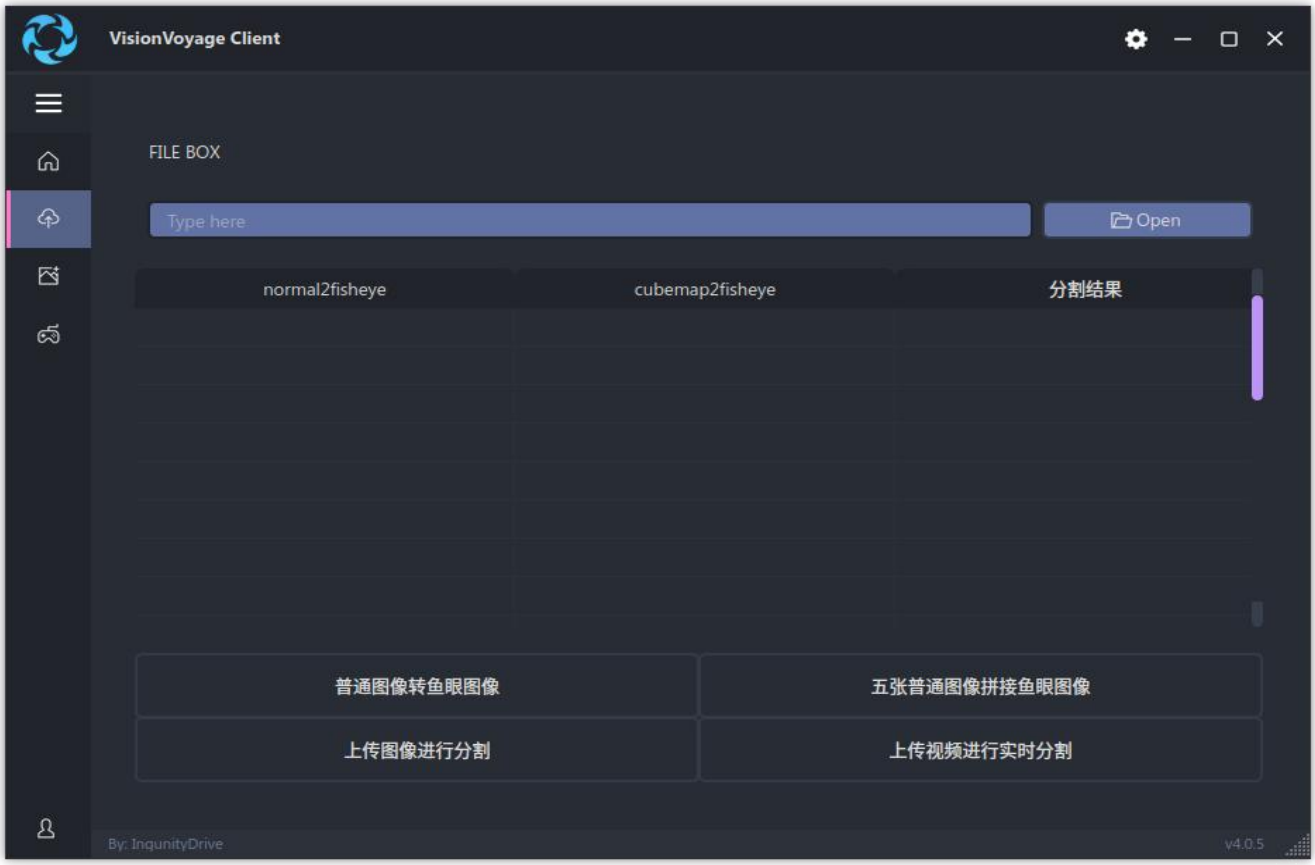

图5.4 上传界面

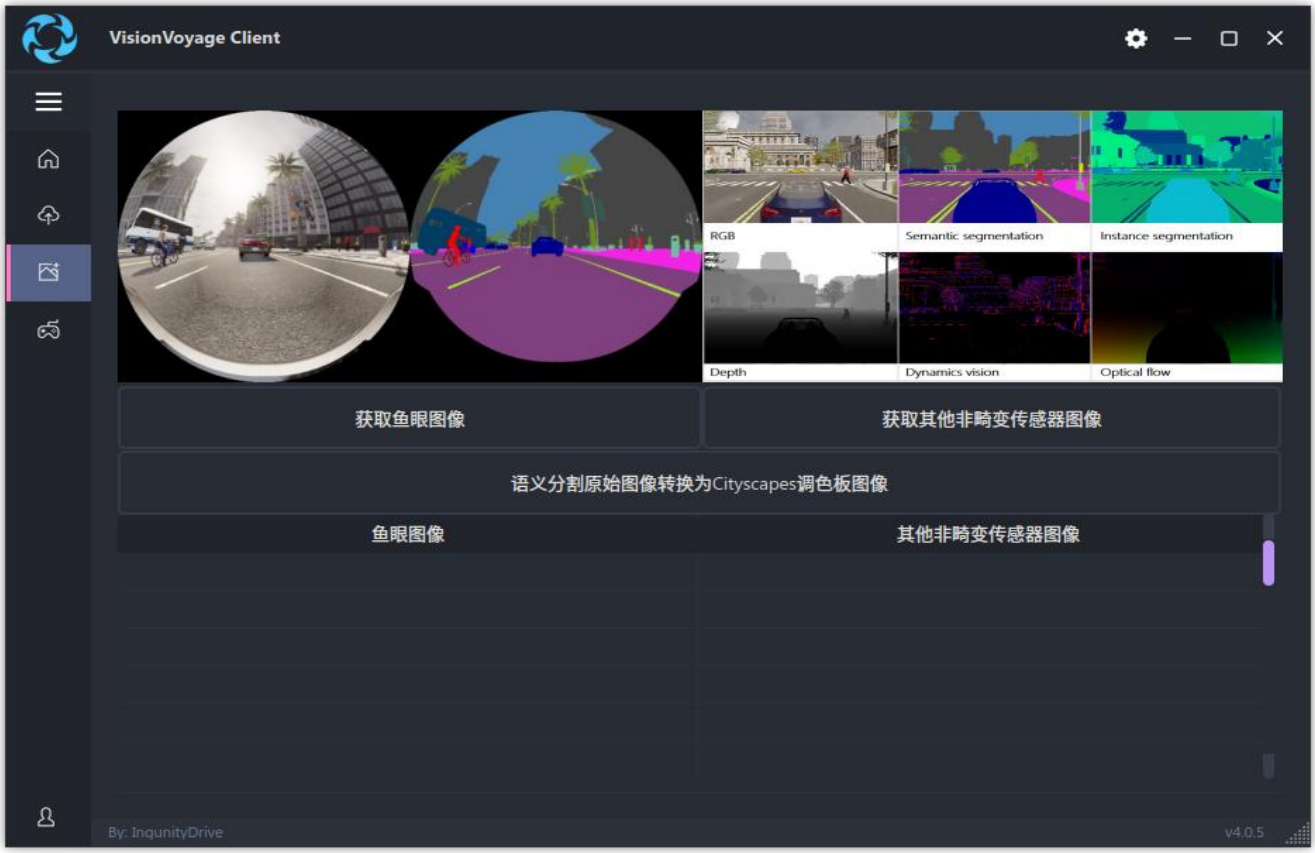

图5.5 获取图像界面


VisionVoyage Client
快捷键操作说明
虚拟驾驶
自动驾驶
按键
作用
按键
作用
W/S
前进/刹车
TAB
改变传感器位置
A/D
左/右转向
N
下一个传感器
Space
手刹
F1
切换HUD显示
TAB
改变传感器位置
R
记录图像到磁盘
N
下一个传感器
[0-9]
切换至传感器[1-9]
F1
切换HUD显示
R
记录图像到磁盘
C
改变天气
V
选择图层
B/SHIFT+B
加载/卸载图层
交通初始化
虚拟驾驶
自动驾驶
By: InqunityDrive
v4.0.5

图5.6 虚拟驾驶界面