郑 州 大 学

生 产 实 习 报 告

专　业　计算机科学与技术

班　级　计科 4 班

学　号　202124100505

姓　名　冯帅迪

计算机与人工智能学院

2024 年 7 月

实习时间	2024 年 6 月 27 日至 7 月 6 日	实习地点	3 楼机房
实习单位	东方瑞通（郑州）咨询服务有限公司		
指导教师	梁鹏		
项目名称	VisionVoyage-基于鱼眼相机与其他感知技术的自动驾驶仿真系统		

实习目的：

1. 提高团队协作能力：在团队中分工合作，使用版本控制工具（如 Git），在实验过程中，采用敏捷开发模式（如 Scrum），学习快速迭代、持续反馈和改进的方法，提升学生在动态环境下开发人工智能项目的能力，提升团队协作和沟通能力。

2.了解在人工智能领域的应用： 通过具体实验，熟悉 Python 在人工智能中的广泛应用，包括数据处理、模型训练、结果评估等。

3. 掌握基本的人工智能算法和模型： 学习并实现常见的人工智能算法，如线性回归、决策树、神经网络等，理解它们的原理和应用场景。

4. 提升 Python 编程能力： 强化对 Python 语言的掌握，尤其是使用 Python 进行数据分析、模型构建和优化的能力，熟悉相关的库和工具（如 NumPy、Pandas、Scikit-learn、TensorFlow、Keras 等）。

5. 熟悉人工智能项目的开发流程： 从数据预处理、模型选择与训练、模型评估与优化，到结果解释与应用，全面了解人工智能项目的完整开发流程。

6. 培养解决实际问题的能力： 通过实验项目中的实际问题，学会分析问题、设计解决方案、实施并验证效果，提升在实际场景中解决问题的能力。

7. 关注人工智能领域的前沿技术和发展趋势： 了解当前人工智能领域的最新研究成果和技术发展趋势，培养创新意识和行业敏锐度。

8. 提升职业素养和专业能力： 通过严格的实验规范和要求，培养职业素养，提升在未来工作中的专业能力和素质。

9. 激发创新意识： 在实验过程中探索新方法、新技术，提出创新性的解决方案，培养独立思考和创新能力。

项目简介：

VisionVoyage 是一款专为自动驾驶领域开发的先进软件，旨在通过利用鱼眼相机及其感知技术，推动自动驾驶技术的发展。该软件解决了当前市场上鱼眼数据集稀缺、鱼眼图像畸变处理难、多摄像头组合语义分割算法等技术瓶颈，填补了行业空白。

主要功能：

1. **普通图像转鱼眼图像**：将常规图像转换为鱼眼图像，模拟真实鱼眼相机的视角。
2. **传感器图像获取与处理**：在仿真环境下获取并处理各种传感器的图像数据。
3. **虚拟驾驶体验**：提供虚拟驾驶环境，让用户体验自动驾驶系统。
4. **自动驾驶仿真**：模拟自动驾驶场景，测试和验证自动驾驶算法。

核心优势：

- **鱼眼相机数据处理**：利用先进的计算机视觉和深度学习技术，实时处理和分析鱼眼相机图像。
- **高精度定位与路径规划**：结合高精度地图数据，实现精准定位和路径规划。
- **多任务语义分割**：针对多摄像头组合设计的语义分割算法，提升感知系统的整体性能。
- **安全性与可靠性**：通过盲区监测和交通环境识别，提高自动驾驶系统的安全性和可靠性。

市场前景： 随着自动驾驶技术的不断成熟和市场需求的增加，VisionVoyage 软件将为研究人员和工程师提供强大的工具，帮助他们更好地理解和应用鱼眼相机和其他传感器数据，提升自动驾驶系统的感知能力和安全性。我们致力于推动自动驾驶技术的发展，满足公众对更先进、更安全交通技术的迫切需求。

VisionVoyage 软件不仅是自动驾驶技术研究的重要工具，也是行业创新的催化剂。通过提供丰富的实验和模拟环境，我们将助力自动驾驶领域的持续进步和创新。

个人承担工作：

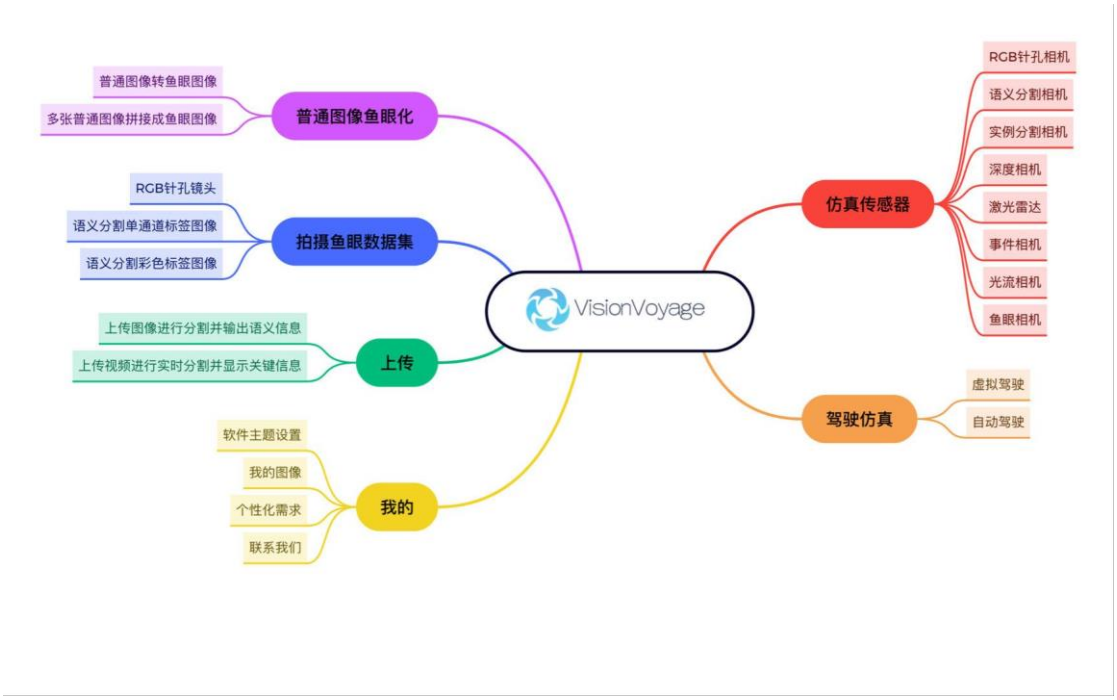
我在项目中承担了部分项目关键算法、功能的实现：

1.软件 UI 编写

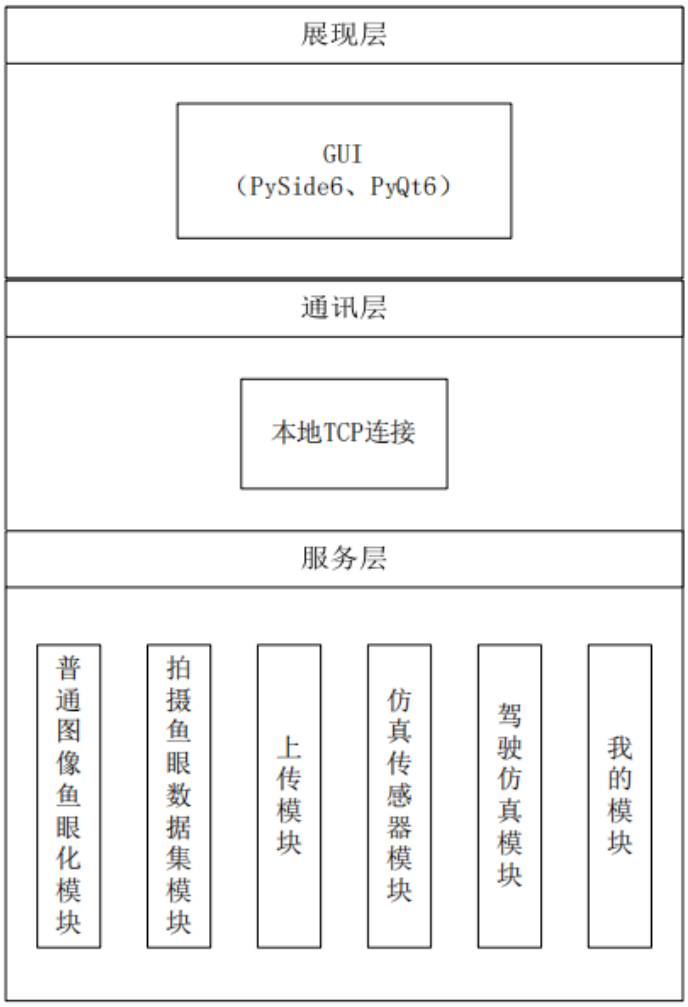
2.交通生成

3.作为产品经理负责产品宣发

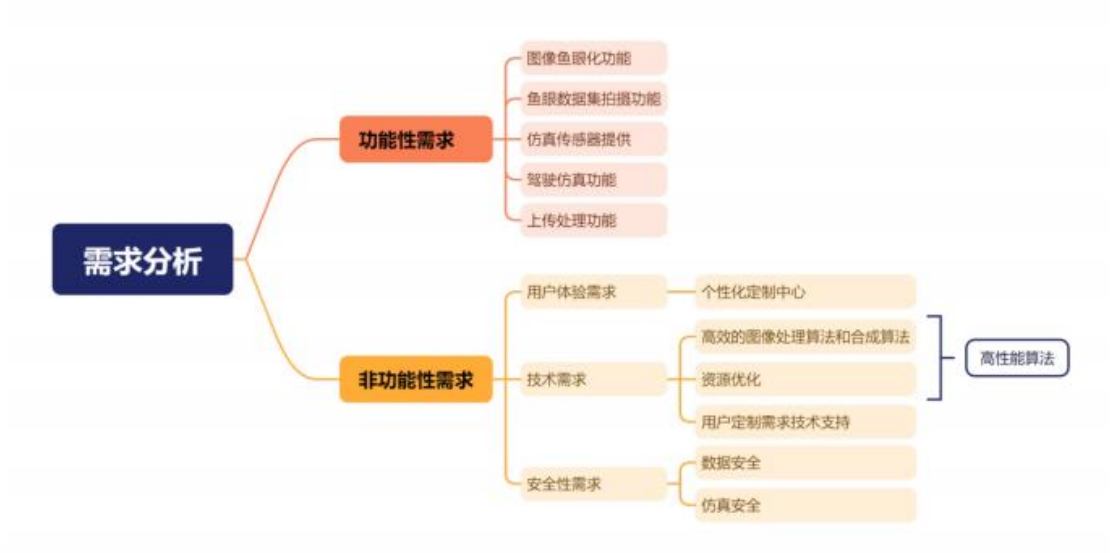
整体功能图：


VisionVoyage
普通图像鱼眼化
普通图像转鱼眼图像
多张普通图像拼接成鱼眼图像
拍摄鱼眼数据集
RGB针孔镜头
语义分割单通道标签图像
语义分割彩色标签图像
上传
上传图像进行分割并输出语义信息
上传视频进行实时分割并显示关键信息
我的
软件主题设置
我的图像
个性化需求
联系我们
仿真传感器
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
驾驶仿真
虚拟驾驶
自动驾驶

逻辑结构图：


展现层
GUI
（PySide6、PyQt6）
通讯层
本地TCP连接
服务层
普通图像鱼眼化模块
拍摄鱼眼数据集模块
上传模块
仿真传感器模块
驾驶仿真模块
我的模块

需求分析图：



重要代码以及主要程序接口：

1.软件 UI 编写（部分代码）：

```
from PySide6.QtCore import (QCoreApplication, QDate, QDateTime,
QLocale,
                            QMetaObject, QObject, QPoint, QRect,
                            QSize, QTime, QUrl, Qt)
from PySide6.QtGui import (QBrush, QColor, QConicalGradient, QCursor,
                           QFont, QFontDatabase, QGradient, QIcon,
                           QImage, QKeySequence, QLinearGradient,
QPainter,
                           QPalette, QPixmap, QRadialGradient,
QTransform)
from PySide6.QtWidgets import (QAbstractItemView,
QAbstractScrollArea, QApplication, QFrame,
                               QGridLayout, QHBoxLayout,
QHeaderView, QLabel,
                               QLineEdit, QMainWindow, QPushButton,
QSizePolicy,
                               QStackedWidget, QTableWidget,
QTableWidgetItem, QTextEdit,
                               QVBoxLayout, QWidget)
import resources_rc
import resources_rc
import resources_rc


class Ui_MainWindow(object):
```

```
    def setupUi(self, MainWindow):
        if not MainWindow.objectName():
            MainWindow.setObjectName(u"MainWindow")
        MainWindow.resize(947, 686)
        MainWindow.setMinimumSize(QSize(940, 560))
        font = QFont()
        font.setFamilies([u"Segoe UI"])
        MainWindow.setFont(font)
        self.styleSheet = QWidget(MainWindow)
        self.styleSheet.setObjectName(u"styleSheet")
        font1 = QFont()
        font1.setFamilies([u"Segoe UI"])
        font1.setPointSize(10)
        font1.setBold(False)
        font1.setItalic(False)
        self.styleSheet.setFont(font1)
        self.appMargins = QVBoxLayout(self.styleSheet)
        self.appMargins.setSpacing(0)
        self.appMargins.setObjectName(u"appMargins")
        self.appMargins.setContentsMargins(10, 10, 10, 10)
        self.bgApp = QFrame(self.styleSheet)
        self.bgApp.setObjectName(u"bgApp")
        self.bgApp.setStyleSheet(u"")
        self.bgApp.setFrameShape(QFrame.NoFrame)
        self.bgApp.setFrameShadow(QFrame.Raised)
        self.appLayout = QHBoxLayout(self.bgApp)
        self.appLayout.setSpacing(0)
        self.appLayout.setObjectName(u"appLayout")
        self.appLayout.setContentsMargins(0, 0, 0, 0)
        self.leftMenuBg = QFrame(self.bgApp)
        self.leftMenuBg.setObjectName(u"leftMenuBg")
        self.leftMenuBg.setMinimumSize(QSize(60, 0))
        self.leftMenuBg.setMaximumSize(QSize(60, 16777215))
        self.leftMenuBg.setFrameShape(QFrame.NoFrame)
        self.leftMenuBg.setFrameShadow(QFrame.Raised)
        self.verticalLayout_3 = QVBoxLayout(self.leftMenuBg)
        self.verticalLayout_3.setSpacing(0)
        self.verticalLayout_3.setObjectName(u"verticalLayout_3")
        self.verticalLayout_3.setContentsMargins(0, 0, 0, 0)
        self.topLogoInfo = QFrame(self.leftMenuBg)
        self.topLogoInfo.setObjectName(u"topLogoInfo")
        self.topLogoInfo.setMinimumSize(QSize(0, 50))
        self.topLogoInfo.setMaximumSize(QSize(16777215, 50))
        self.topLogoInfo.setFrameShape(QFrame.NoFrame)
```

```
        self.topLogoInfo.setFrameShadow(QFrame.Raised)
        self.topLogo = QFrame(self.topLogoInfo)
        self.topLogo.setObjectName(u"topLogo")
        self.topLogo.setGeometry(QRect(10, 5, 42, 42))
        self.topLogo.setMinimumSize(QSize(42, 42))
        self.topLogo.setMaximumSize(QSize(42, 42))
        self.topLogo.setFrameShape(QFrame.NoFrame)
        self.topLogo.setFrameShadow(QFrame.Raised)
        self.titleLeftApp = QLabel(self.topLogoInfo)
        self.titleLeftApp.setObjectName(u"titleLeftApp")
        self.titleLeftApp.setGeometry(QRect(70, 8, 160, 20))
        font2 = QFont()
        font2.setFamilies([u"Segoe UI"])
        font2.setPointSize(8)
        font2.setBold(False)
        font2.setItalic(False)
        self.titleLeftApp.setFont(font2)
        self.titleLeftApp.setAlignment(Qt.AlignLeading |
Qt.AlignLeft | Qt.AlignTop)
        self.titleLeftDescription = QLabel(self.topLogoInfo)

self.titleLeftDescription.setObjectName(u"titleLeftDescription")
        self.titleLeftDescription.setGeometry(QRect(70, 27, 160, 16))
        self.titleLeftDescription.setMaximumSize(QSize(16777215,
16))
        self.titleLeftDescription.setFont(font2)
        self.titleLeftDescription.setAlignment(Qt.AlignLeading |
Qt.AlignLeft | Qt.AlignTop)

        self.verticalLayout_3.addWidget(self.topLogoInfo)

        self.leftMenuFrame = QFrame(self.leftMenuBg)
        self.leftMenuFrame.setObjectName(u"leftMenuFrame")
        self.leftMenuFrame.setFrameShape(QFrame.NoFrame)
        self.leftMenuFrame.setFrameShadow(QFrame.Raised)
        self.verticalMenuLayout = QVBoxLayout(self.leftMenuFrame)
        self.verticalMenuLayout.setSpacing(0)

self.verticalMenuLayout.setObjectName(u"verticalMenuLayout")
        self.verticalMenuLayout.setContentsMargins(0, 0, 0, 0)
        self.toggleBox = QFrame(self.leftMenuFrame)
        self.toggleBox.setObjectName(u"toggleBox")
        self.toggleBox.setMaximumSize(QSize(16777215, 45))
        self.toggleBox.setFrameShape(QFrame.NoFrame)
```

```
        self.toggleBox.setFrameShadow(QFrame.Raised)
        self.verticalLayout_4 = QVBoxLayout(self.toggleBox)
        self.verticalLayout_4.setSpacing(0)
        self.verticalLayout_4.setObjectName(u"verticalLayout_4")
        self.verticalLayout_4.setContentsMargins(0, 0, 0, 0)
        self.toggleButton = QPushButton(self.toggleBox)
        self.toggleButton.setObjectName(u"toggleButton")
        sizePolicy = QSizePolicy(QSizePolicy.Expanding,
QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.toggleButton.sizePolicy().hasHeig
htForWidth())
        self.toggleButton.setSizePolicy(sizePolicy)
        self.toggleButton.setMinimumSize(QSize(0, 45))
        self.toggleButton.setFont(font1)
        self.toggleButton.setCursor(QCursor(Qt.PointingHandCursor))
        self.toggleButton.setLayoutDirection(Qt.LeftToRight)
        self.toggleButton.setStyleSheet(u"background-image:
url(:/icons/images/icons/icon_menu.png);")

        self.verticalLayout_4.addWidget(self.toggleButton)

        self.verticalMenuLayout.addWidget(self.toggleBox)

        self.topMenu = QFrame(self.leftMenuFrame)
        self.topMenu.setObjectName(u"topMenu")
        self.topMenu.setFrameShape(QFrame.NoFrame)
        self.topMenu.setFrameShadow(QFrame.Raised)
        self.verticalLayout_8 = QVBoxLayout(self.topMenu)
        self.verticalLayout_8.setSpacing(0)
        self.verticalLayout_8.setObjectName(u"verticalLayout_8")
        self.verticalLayout_8.setContentsMargins(0, 0, 0, 0)
        self.btn_home = QPushButton(self.topMenu)
        self.btn_home.setObjectName(u"btn_home")

sizePolicy.setHeightForWidth(self.btn_home.sizePolicy().hasHeightFo
rWidth())
        self.btn_home.setSizePolicy(sizePolicy)
        self.btn_home.setMinimumSize(QSize(0, 45))
        self.btn_home.setFont(font1)
        self.btn_home.setCursor(QCursor(Qt.PointingHandCursor))
        self.btn_home.setLayoutDirection(Qt.LeftToRight)
```

```
        self.btn_home.setStyleSheet(u"background-image:
url(:/icons/images/icons/cil-home.png);")

        self.verticalLayout_8.addWidget(self.btn_home)

        self.btn_widgets = QPushButton(self.topMenu)
        self.btn_widgets.setObjectName(u"btn_widgets")

sizePolicy.setHeightForWidth(self.btn_widgets.sizePolicy().hasHeigh
tForWidth())
        self.btn_widgets.setSizePolicy(sizePolicy)
        self.btn_widgets.setMinimumSize(QSize(0, 45))
        self.btn_widgets.setFont(font1)
        self.btn_widgets.setCursor(QCursor(Qt.PointingHandCursor))
        self.btn_widgets.setLayoutDirection(Qt.LeftToRight)
        self.btn_widgets.setStyleSheet(u"background-image:
url(:/icons/images/icons/cil-cloud-upload.png);")

        self.verticalLayout_8.addWidget(self.btn_widgets)

        self.btn_image = QPushButton(self.topMenu)
        self.btn_image.setObjectName(u"btn_image")

sizePolicy.setHeightForWidth(self.btn_image.sizePolicy().hasHeightF
orWidth())
        self.btn_image.setSizePolicy(sizePolicy)
        self.btn_image.setMinimumSize(QSize(0, 45))
        self.btn_image.setFont(font1)
        self.btn_image.setCursor(QCursor(Qt.PointingHandCursor))
        self.btn_image.setLayoutDirection(Qt.LeftToRight)
        self.btn_image.setStyleSheet(u"background-image:
url(:/icons/images/icons/cil-image-plus.png);")

        self.verticalLayout_8.addWidget(self.btn_image)

        self.btn_simulation = QPushButton(self.topMenu)
        self.btn_simulation.setObjectName(u"btn_simulation")

sizePolicy.setHeightForWidth(self.btn_simulation.sizePolicy().hasHe
ightForWidth())
        self.btn_simulation.setSizePolicy(sizePolicy)
        self.btn_simulation.setMinimumSize(QSize(0, 45))
        self.btn_simulation.setFont(font1)
```

```
self.btn_simulation.setCursor(QCursor(Qt.PointingHandCursor))
        self.btn_simulation.setLayoutDirection(Qt.LeftToRight)
        self.btn_simulation.setStyleSheet(u"background-image:
url(:/icons/images/icons/cil-gamepad.png);")

        self.verticalLayout_8.addWidget(self.btn_simulation)

        self.verticalMenuLayout.addWidget(self.topMenu, 0,
Qt.AlignTop)

        self.bottomMenu = QFrame(self.leftMenuFrame)
        self.bottomMenu.setObjectName(u"bottomMenu")
        self.bottomMenu.setFrameShape(QFrame.NoFrame)
        self.bottomMenu.setFrameShadow(QFrame.Raised)
        self.verticalLayout_9 = QVBoxLayout(self.bottomMenu)
        self.verticalLayout_9.setSpacing(0)
        self.verticalLayout_9.setObjectName(u"verticalLayout_9")
        self.verticalLayout_9.setContentsMargins(0, 0, 0, 0)
        self.btn_personal_center = QPushButton(self.bottomMenu)

self.btn_personal_center.setObjectName(u"btn_personal_center")

sizePolicy.setHeightForWidth(self.btn_personal_center.sizePolicy().
hasHeightForWidth())
        self.btn_personal_center.setSizePolicy(sizePolicy)
        self.btn_personal_center.setMinimumSize(QSize(0, 45))
        self.btn_personal_center.setFont(font1)

self.btn_personal_center.setCursor(QCursor(Qt.PointingHandCursor))
        self.btn_personal_center.setLayoutDirection(Qt.LeftToRight)
        self.btn_personal_center.setStyleSheet(u"background-image:
url(:/icons/images/icons/cil-user.png);")

        self.verticalLayout_9.addWidget(self.btn_personal_center)

        self.verticalMenuLayout.addWidget(self.bottomMenu, 0,
Qt.AlignBottom)

        self.verticalLayout_3.addWidget(self.leftMenuFrame)

        self.appLayout.addWidget(self.leftMenuBg)

        self.extraLeftBox = QFrame(self.bgApp)
        self.extraLeftBox.setObjectName(u"extraLeftBox")
```

```
        self.extraLeftBox.setMinimumSize(QSize(0, 0))
        self.extraLeftBox.setMaximumSize(QSize(0, 16777215))
        self.extraLeftBox.setFrameShape(QFrame.NoFrame)
        self.extraLeftBox.setFrameShadow(QFrame.Raised)
        self.extraColumLayout = QVBoxLayout(self.extraLeftBox)
        self.extraColumLayout.setSpacing(0)
        self.extraColumLayout.setObjectName(u"extraColumLayout")
        self.extraColumLayout.setContentsMargins(0, 0, 0, 0)
        self.extraTopBg = QFrame(self.extraLeftBox)
        self.extraTopBg.setObjectName(u"extraTopBg")
        self.extraTopBg.setMinimumSize(QSize(0, 50))
        self.extraTopBg.setMaximumSize(QSize(16777215, 50))
        self.extraTopBg.setFrameShape(QFrame.NoFrame)
        self.extraTopBg.setFrameShadow(QFrame.Raised)
        self.verticalLayout_5 = QVBoxLayout(self.extraTopBg)
        self.verticalLayout_5.setSpacing(0)
        self.verticalLayout_5.setObjectName(u"verticalLayout_5")
        self.verticalLayout_5.setContentsMargins(0, 0, 0, 0)
        self.extraTopLayout = QGridLayout()
        self.extraTopLayout.setObjectName(u"extraTopLayout")
        self.extraTopLayout.setHorizontalSpacing(10)
        self.extraTopLayout.setVerticalSpacing(0)
        self.extraTopLayout.setContentsMargins(10, -1, 10, -1)
        self.extraIcon = QFrame(self.extraTopBg)
        self.extraIcon.setObjectName(u"extraIcon")
        self.extraIcon.setMinimumSize(QSize(20, 0))
        self.extraIcon.setMaximumSize(QSize(20, 20))
        self.extraIcon.setFrameShape(QFrame.NoFrame)
        self.extraIcon.setFrameShadow(QFrame.Raised)

        self.extraTopLayout.addWidget(self.extraIcon, 0, 0, 1, 1)

        self.extraLabel = QLabel(self.extraTopBg)
        self.extraLabel.setObjectName(u"extraLabel")
        self.extraLabel.setMinimumSize(QSize(150, 0))

        self.extraTopLayout.addWidget(self.extraLabel, 0, 1, 1, 1)

        self.extraCloseColumnBtn = QPushButton(self.extraTopBg)

self.extraCloseColumnBtn.setObjectName(u"extraCloseColumnBtn")
        self.extraCloseColumnBtn.setMinimumSize(QSize(28, 28))
        self.extraCloseColumnBtn.setMaximumSize(QSize(28, 28))
```

```
self.extraCloseColumnBtn.setCursor(QCursor(Qt.PointingHandCursor))
        self.extraCloseColumnBtn.setStyleSheet(u"")
        icon = QIcon()
        icon.addFile(u":/icons/images/icons/icon_close.png",
QSize(), QIcon.Normal, QIcon.Off)
        self.extraCloseColumnBtn.setIcon(icon)
        self.extraCloseColumnBtn.setIconSize(QSize(20, 20))

        self.extraTopLayout.addWidget(self.extraCloseColumnBtn, 0,
2, 1, 1)

        self.verticalLayout_5.addLayout(self.extraTopLayout)

        self.extraColumLayout.addWidget(self.extraTopBg)
```

2.交通生成：

```
def get_actor_blueprints(world, filter, generation):
    bps = world.get_blueprint_library().filter(filter)

    if generation.lower() == "all":
        return bps

    # If the filter returns only one bp, we assume that this one needed
    # and therefore, we ignore the generation
    if len(bps) == 1:
        return bps

    try:
        int_generation = int(generation)
        # Check if generation is in available generations
        if int_generation in [1, 2]:
            bps = [x for x in bps if int(x.get_attribute('generation'))
== int_generation]
            return bps
        else:
            print("   Warning! Actor Generation is not valid. No actor
will be spawned.")
            return []
    except:
        print("   Warning! Actor Generation is not valid. No actor will
be spawned.")
        return []
```

```
def main():
    argparser = argparse.ArgumentParser(
        description=__doc__)
    argparser.add_argument(
        '--host',
        metavar='H',
        default='127.0.0.1',
        help='IP of the host server (default: 127.0.0.1)')
    argparser.add_argument(
        '-p', '--port',
        metavar='P',
        default=2000,
        type=int,
        help='TCP port to listen to (default: 2000)')
    argparser.add_argument(
        '-n', '--number-of-vehicles',
        metavar='N',
        default=50,
        type=int,
        help='Number of vehicles (default: 30)')
    argparser.add_argument(
        '-w', '--number-of-walkers',
        metavar='W',
        default=50,
        type=int,
        help='Number of walkers (default: 10)')
    argparser.add_argument(
        '--safe',
        action='store_true',
        help='Avoid spawning vehicles prone to accidents')
    argparser.add_argument(
        '--filterv',
        metavar='PATTERN',
        default='vehicle.*',
        help='Filter vehicle model (default: "vehicle.*")')
    argparser.add_argument(
        '--generationv',
        metavar='G',
        default='All',
        help='restrict to certain vehicle generation (values:
"1","2","All" - default: "All")')
    argparser.add_argument(
        '--filterw',
```

```
        metavar='PATTERN',
        default='walker.pedestrian.*',
        help='Filter pedestrian type (default:
"walker.pedestrian.*")')
    argparser.add_argument(
        '--generationw',
        metavar='G',
        default='2',
        help='restrict to certain pedestrian generation (values:
"1","2","All" - default: "2")')
    argparser.add_argument(
        '--tm-port',
        metavar='P',
        default=8000,
        type=int,
        help='Port to communicate with TM (default: 8000)')
    argparser.add_argument(
        '--asynch',
        action='store_true',
        help='Activate asynchronous mode execution')
    argparser.add_argument(
        '--hybrid',
        action='store_true',
        help='Activate hybrid mode for Traffic Manager')
    argparser.add_argument(
        '-s', '--seed',
        metavar='S',
        type=int,
        help='Set random device seed and deterministic mode for Traffic
Manager')
    argparser.add_argument(
        '--seedw',
        metavar='S',
        default=0,
        type=int,
        help='Set the seed for pedestrians module')
    argparser.add_argument(
        '--car-lights-on',
        action='store_true',
        default=False,
        help='Enable automatic car light management')
    argparser.add_argument(
        '--hero',
        action='store_true',
```

```
        default=False,
        help='Set one of the vehicles as hero')
    argparser.add_argument(
        '--respawn',
        action='store_true',
        default=False,
        help='Automatically respawn dormant vehicles (only in large
maps)')
    argparser.add_argument(
        '--no-rendering',
        action='store_true',
        default=False,
        help='Activate no rendering mode')

    args = argparser.parse_args()

    logging.basicConfig(format='%(levelname)s: %(message)s',
level=logging.INFO)

    vehicles_list = []
    walkers_list = []
    all_id = []
    client = carla.Client(args.host, args.port)
    client.set_timeout(10.0)
    synchronous_master = False
    random.seed(args.seed if args.seed is not None else
int(time.time()))

    try:
        world = client.get_world()

        traffic_manager = client.get_trafficmanager(args.tm_port)
        traffic_manager.set_global_distance_to_leading_vehicle(2.5)
        if args.respawn:
            traffic_manager.set_respawn_dormant_vehicles(True)
        if args.hybrid:
            traffic_manager.set_hybrid_physics_mode(True)
            traffic_manager.set_hybrid_physics_radius(70.0)
        if args.seed is not None:
            traffic_manager.set_random_device_seed(args.seed)

        settings = world.get_settings()
        if not args.asynch:
            traffic_manager.set_synchronous_mode(True)
```

```
            if not settings.synchronous_mode:
                synchronous_master = True
                settings.synchronous_mode = True
                settings.fixed_delta_seconds = 0.05
            else:
                synchronous_master = False
        else:
            print("You are currently in asynchronous mode. If this is
a traffic simulation, \
            you could experience some issues. If it's not working
correctly, switch to synchronous \
            mode by using
traffic_manager.set_synchronous_mode(True)")

        if args.no_rendering:
            settings.no_rendering_mode = True
        world.apply_settings(settings)

        blueprints = get_actor_blueprints(world, args.filterv,
args.generationv)
        blueprintsWalkers = get_actor_blueprints(world, args.filterw,
args.generationw)

        if args.safe:
            blueprints = [x for x in blueprints if
x.get_attribute('base_type') == 'car']

        blueprints = sorted(blueprints, key=lambda bp: bp.id)

        filtered_list = [bp for bp in blueprints if not
fnmatch.fnmatch(bp.id, 'vehicle.tesla.*')]

        # filtered_blueprints_two_wheels = [bp for bp in filtered_list
if any(fnmatch.fnmatch(bp.id, pattern) for pattern in [
        #     "vehicle.diamondback.century",
"vehicle.gazelle.omafiets", "vehicle.harley-davidson.low_rider",
"vehicle.yamaha.yzf"])]

        spawn_points = world.get_map().get_spawn_points()
        number_of_spawn_points = len(spawn_points)

        if args.number_of_vehicles < number_of_spawn_points:
            random.shuffle(spawn_points)
        elif args.number_of_vehicles > number_of_spawn_points:
```

```
            msg = 'requested %d vehicles, but could only find %d spawn
points'
            logging.warning(msg, args.number_of_vehicles,
number_of_spawn_points)
            args.number_of_vehicles = number_of_spawn_points

        # @todo cannot import these directly.
        SpawnActor = carla.command.SpawnActor
        SetAutopilot = carla.command.SetAutopilot
        FutureActor = carla.command.FutureActor

        # --------------
        # Spawn vehicles
        # --------------
        batch = []
        hero = args.hero
        for n, transform in enumerate(spawn_points):
            if n >= args.number_of_vehicles:
                break
            blueprint = random.choice(filtered_list)
            if blueprint.has_attribute('color'):
                color =
random.choice(blueprint.get_attribute('color').recommended_values)
                blueprint.set_attribute('color', color)
            if blueprint.has_attribute('driver_id'):
                driver_id =
random.choice(blueprint.get_attribute('driver_id').recommended_valu
es)
                blueprint.set_attribute('driver_id', driver_id)
            if hero:
                blueprint.set_attribute('role_name', 'hero')
                hero = False
            else:
                blueprint.set_attribute('role_name', 'autopilot')

            # spawn the cars and set their autopilot and light state
all together
            batch.append(SpawnActor(blueprint, transform)
                         .then(SetAutopilot(FutureActor, True,
traffic_manager.get_port())))

        for response in client.apply_batch_sync(batch,
synchronous_master):
            if response.error:
```

```
                logging.error(response.error)
            else:
                vehicles_list.append(response.actor_id)

        # Set automatic vehicle lights update if specified
        if args.car_lights_on:
            all_vehicle_actors = world.get_actors(vehicles_list)
            for actor in all_vehicle_actors:
                traffic_manager.update_vehicle_lights(actor, True)

        # -------------
        # Spawn Walkers
        # -------------
        # some settings
        percentagePedestriansRunning = 0.0      # how many pedestrians
will run
        percentagePedestriansCrossing = 0.0     # how many pedestrians
will walk through the road
        if args.seedw:
            world.set_pedestrians_seed(args.seedw)
            random.seed(args.seedw)
        # 1. take all the random locations to spawn
        spawn_points = []
        for i in range(args.number_of_walkers):
            spawn_point = carla.Transform()
            loc = world.get_random_location_from_navigation()
            if (loc != None):
                spawn_point.location = loc
                spawn_points.append(spawn_point)
        # 2. we spawn the walker object
        batch = []
        walker_speed = []
        for spawn_point in spawn_points:
            walker_bp = random.choice(blueprintsWalkers)
            # set as not invincible
            if walker_bp.has_attribute('is_invincible'):
                walker_bp.set_attribute('is_invincible', 'false')
            # set the max speed
            if walker_bp.has_attribute('speed'):
                if (random.random() > percentagePedestriansRunning):
                    # walking

walker_speed.append(walker_bp.get_attribute('speed').recommended_va
lues[1])
```

```
                else:
                    # running

walker_speed.append(walker_bp.get_attribute('speed').recommended_va
lues[2])
            else:
                print("Walker has no speed")
                walker_speed.append(0.0)
            batch.append(SpawnActor(walker_bp, spawn_point))
        results = client.apply_batch_sync(batch, True)
        walker_speed2 = []
        for i in range(len(results)):
            if results[i].error:
                logging.error(results[i].error)
            else:
                walkers_list.append({"id": results[i].actor_id})
                walker_speed2.append(walker_speed[i])
        walker_speed = walker_speed2
        # 3. we spawn the walker controller
        batch = []
        walker_controller_bp =
world.get_blueprint_library().find('controller.ai.walker')
        for i in range(len(walkers_list)):
            batch.append(SpawnActor(walker_controller_bp,
carla.Transform(), walkers_list[i]["id"]))
        results = client.apply_batch_sync(batch, True)
        for i in range(len(results)):
            if results[i].error:
                logging.error(results[i].error)
            else:
                walkers_list[i]["con"] = results[i].actor_id
        # 4. we put together the walkers and controllers id to get the
objects from their id
        for i in range(len(walkers_list)):
            all_id.append(walkers_list[i]["con"])
            all_id.append(walkers_list[i]["id"])
        all_actors = world.get_actors(all_id)

        # wait for a tick to ensure client receives the last transform
of the walkers we have just created
        if args.asynch or not synchronous_master:
            world.wait_for_tick()
        else:
            world.tick()
```

```
        # 5. initialize each controller and set target to walk to (list
is [controler, actor, controller, actor ...])
        # set how many pedestrians can cross the road

world.set_pedestrians_cross_factor(percentagePedestriansCrossing)
        for i in range(0, len(all_id), 2):
            # start walker
            all_actors[i].start()
            # set walk to random point

all_actors[i].go_to_location(world.get_random_location_from_navigat
ion())
            # max speed

all_actors[i].set_max_speed(float(walker_speed[int(i/2)]))

        print('spawned %d vehicles and %d walkers, press Ctrl+C to
exit.' % (len(vehicles_list), len(walkers_list)))

        # Example of how to use Traffic Manager parameters
        traffic_manager.global_percentage_speed_difference(30.0)

        while True:
            if not args.asynch and synchronous_master:
                world.tick()
            else:
                world.wait_for_tick()

    finally:

        if not args.asynch and synchronous_master:
            settings = world.get_settings()
            settings.synchronous_mode = False
            settings.no_rendering_mode = False
            settings.fixed_delta_seconds = None
            world.apply_settings(settings)

        print('\ndestroying %d vehicles' % len(vehicles_list))
        client.apply_batch([carla.command.DestroyActor(x) for x in
vehicles_list])

        # stop walker controllers (list is [controller, actor,
controller, actor ...])
```

```
        for i in range(0, len(all_id), 2):
            all_actors[i].stop()

        print('\ndestroying %d walkers' % len(walkers_list))
        client.apply_batch([carla.command.DestroyActor(x) for x in
all_id])

        time.sleep(0.5)
```

3.作为产品经理负责宣发：

我在项目中担任产品经理，负责产品的宣传推广和市场发展战略的制定与实施。

承担的功能和任务

在项目中，我主要负责以下宣发相关的功能和任务：

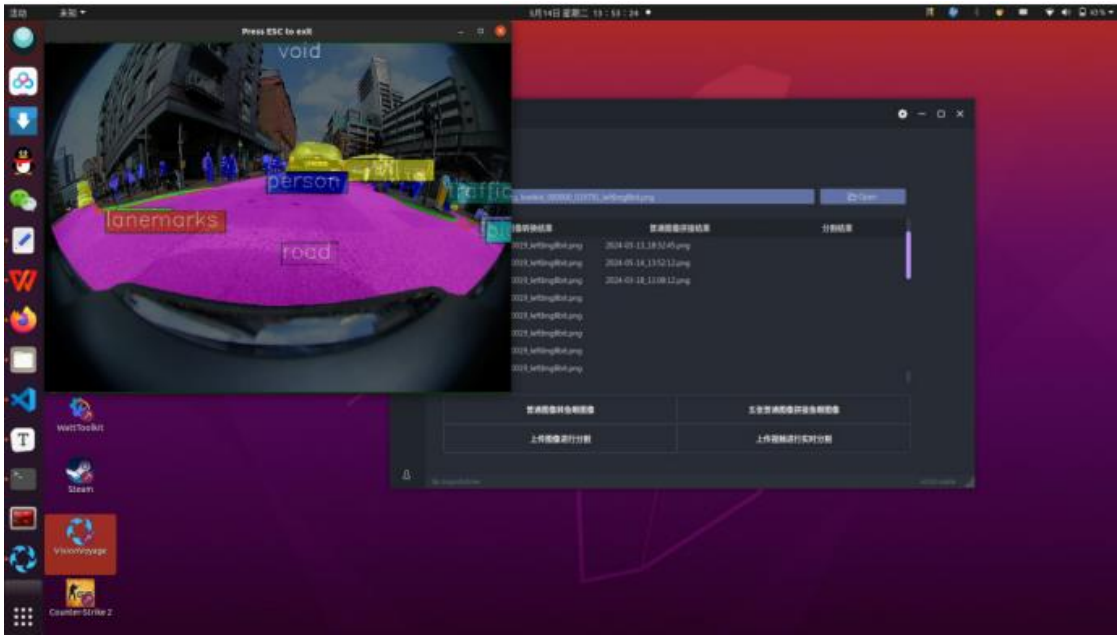
- **市场调研和用户分析**：通过市场调研和用户反馈数据分析，深入了解目标用户群体的需求和行为习惯，为产品定位和推广策略提供数据支持。
- **品牌推广和宣传策略制定**：制定并执行多渠道的品牌推广和宣传策略，包括社交媒体、电子邮件营销、行业展会等，提升产品知名度和用户粘性。
- **合作伙伴关系管理**：与行业内外的合作伙伴建立和维护良好的关系，开展联合营销活动和合作推广，拓展产品的市场影响力和渠道覆盖面。
- **用户反馈和产品优化**：收集和分析用户反馈，与产品团队紧密合作进行产品优化和迭代，提升用户体验和产品竞争力。

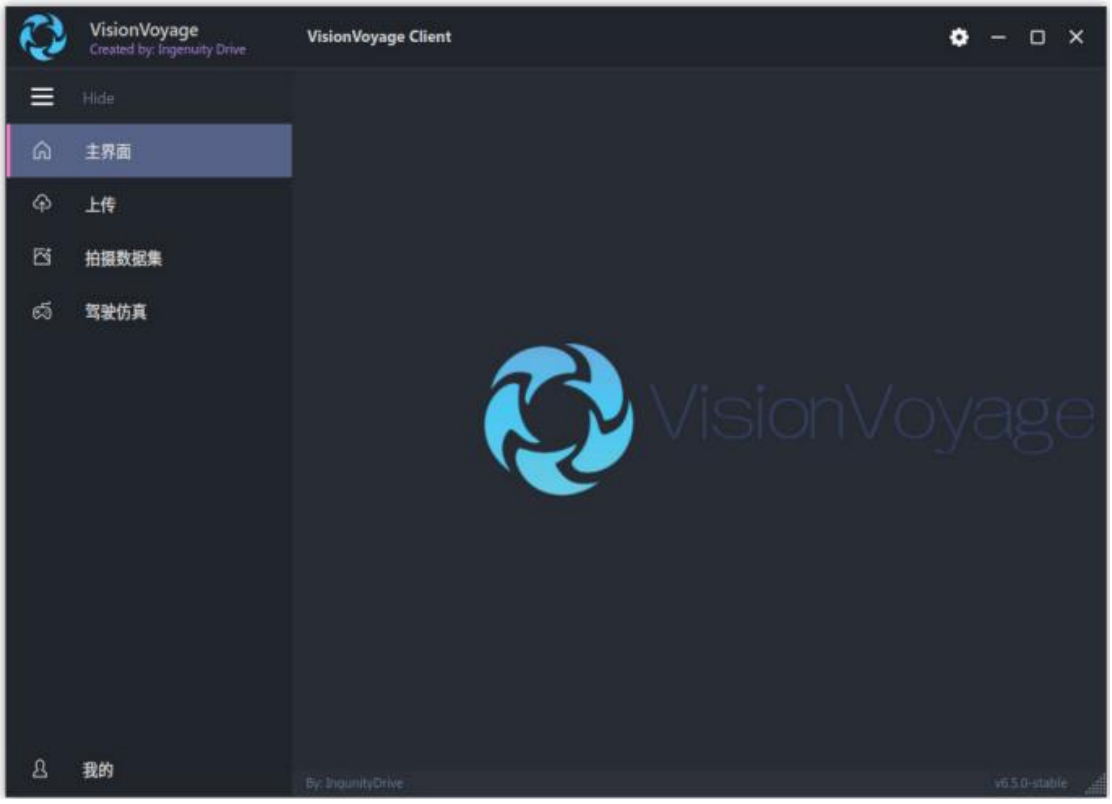
成果和贡献

通过上述工作，成功推广了产品并拓展了用户基础，有效提升了市场份额和品牌价值。具体成就包括但不限于：

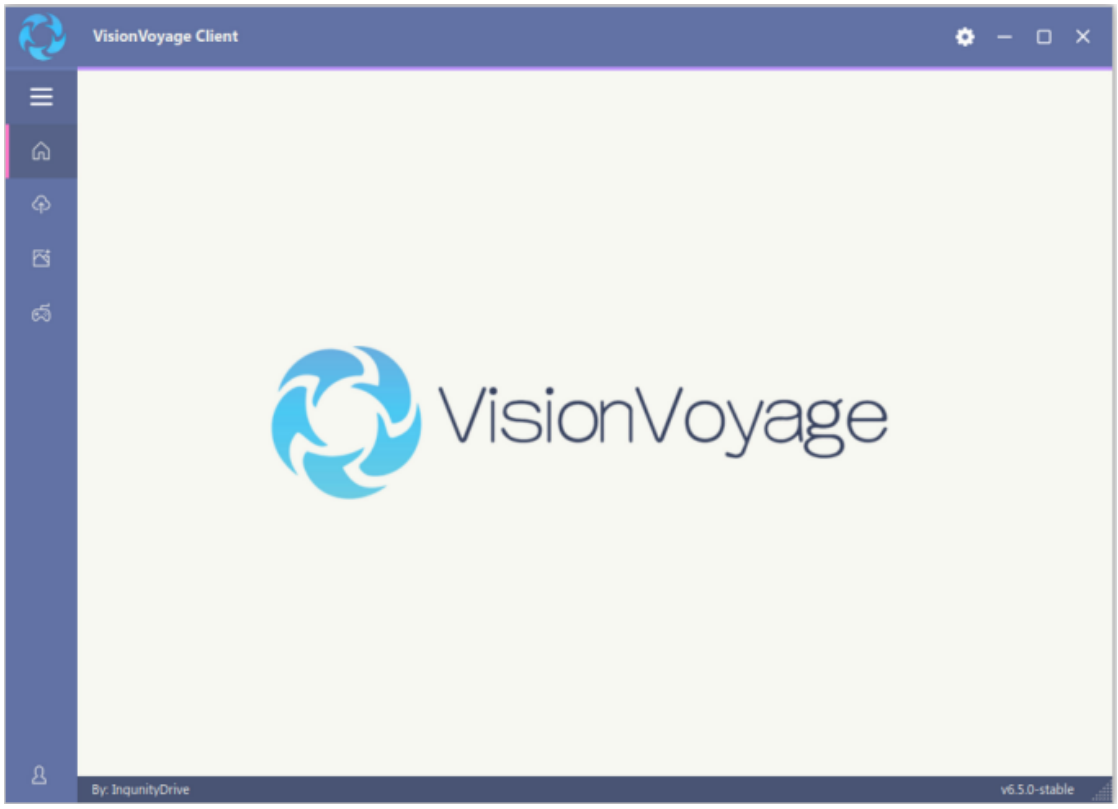
- 增加了用户注册和活跃率，提升了用户参与度和留存率。
- 扩大了产品的市场覆盖范围，进入了新的用户群体和市场领域。
- 提升了品牌在目标市场的知名度和影响力，为公司业务增长和发展奠定了坚实基础。

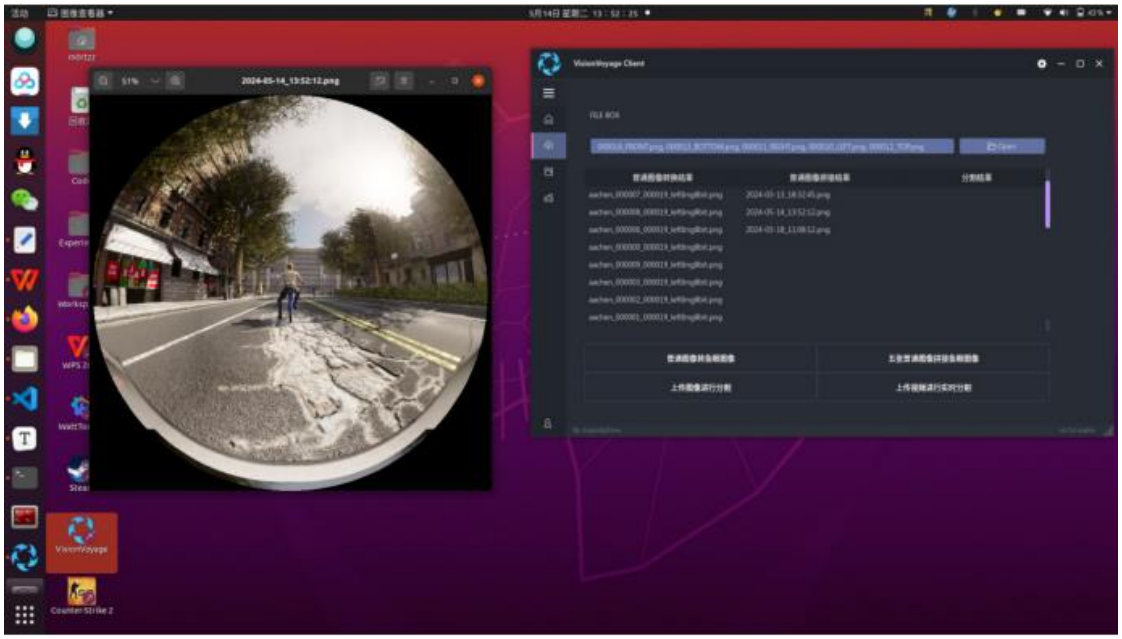
项目效果图：


Press ESC to exit
void
person
lanemarks
road


VisionVoyage
Created by: Ingenuity Drive
VisionVoyage Client
Hide
主界面
上传
拍摄数据集
驾驶仿真
我的
VisionVoyage
By: IngunityDrive
v0.5.0-stable

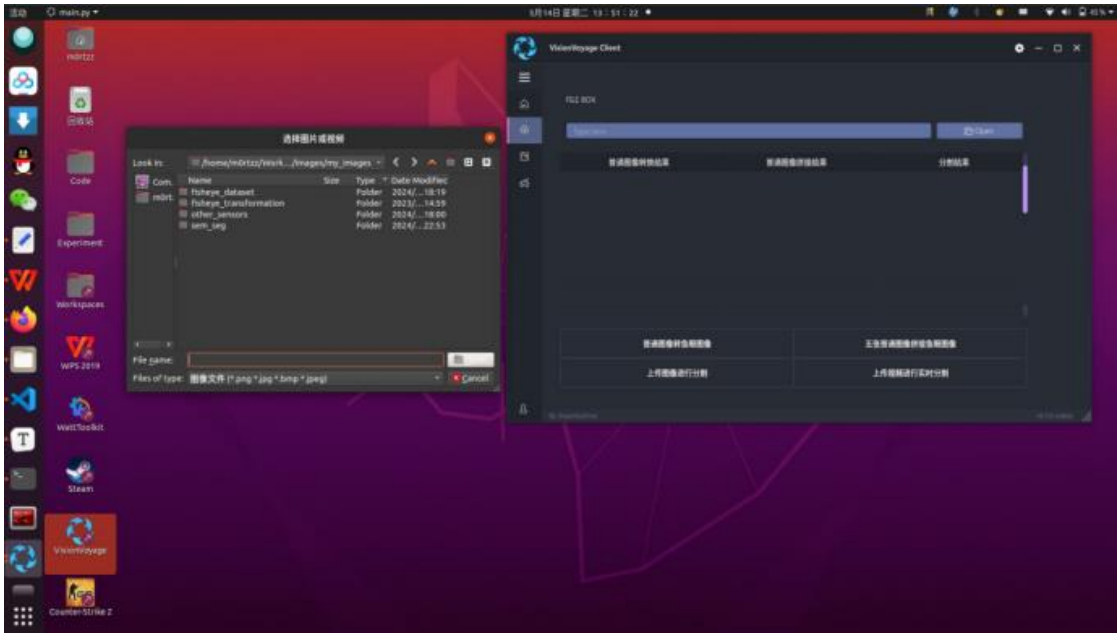

video.mp4
traffic light 0.78
traffic light 0.59
traffic light 0.50
car 0.92
car 0.93
car 0.81
car 0.82

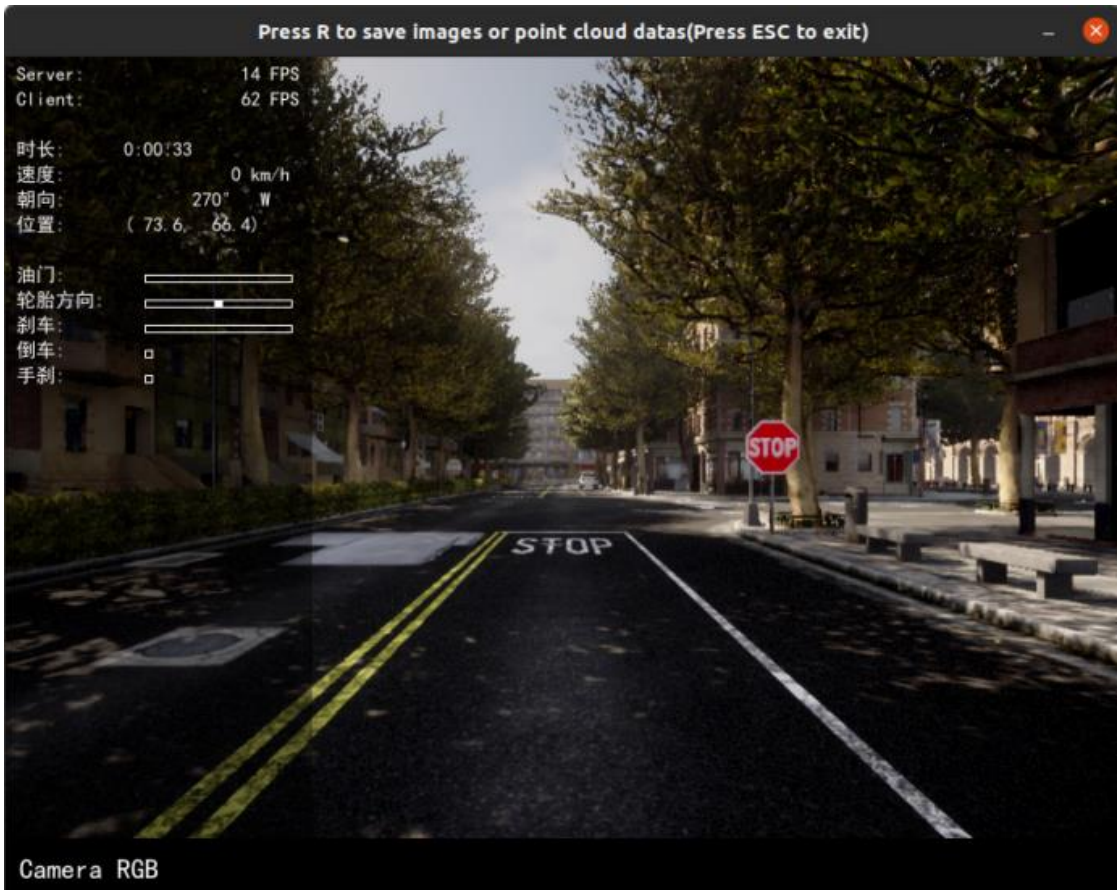

VisionVoyage Client
VisionVoyage
By: InqunityDrive
v6.5.0-stable

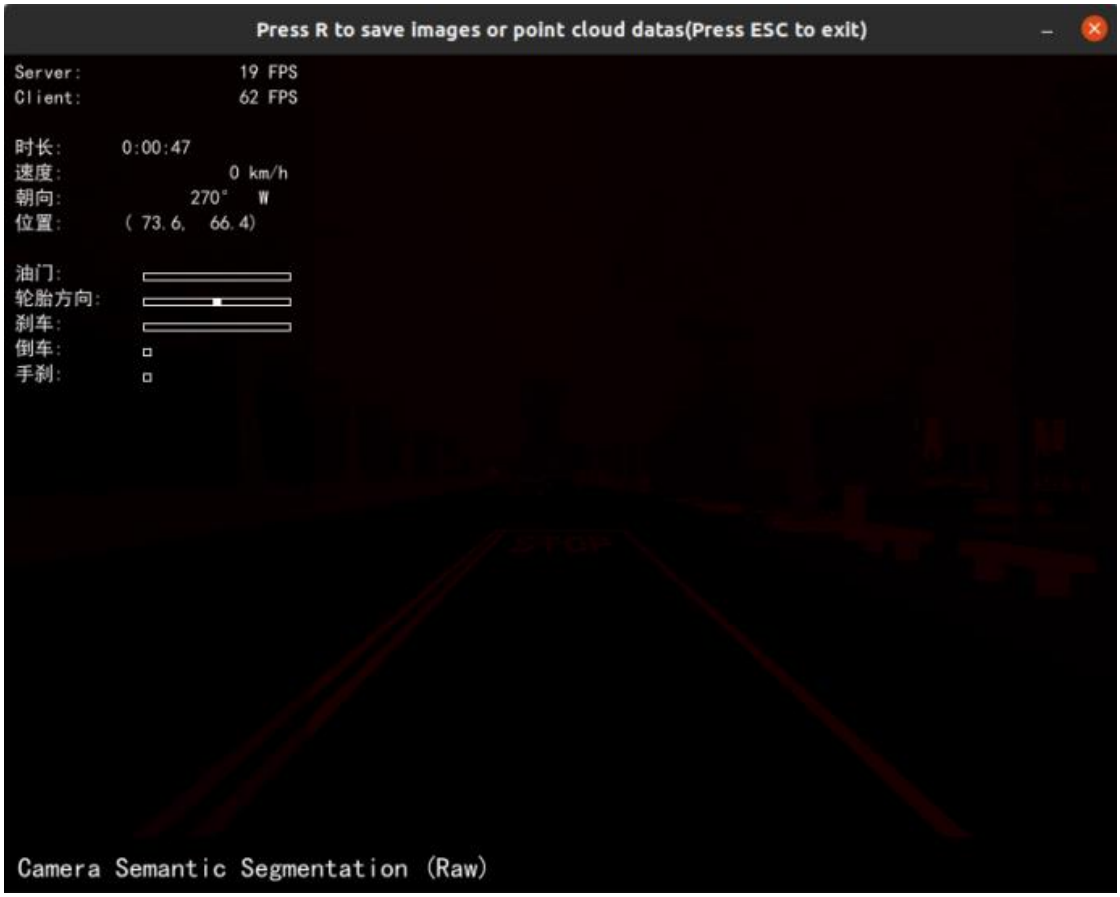

VisionVoyage Client
FILE BOX

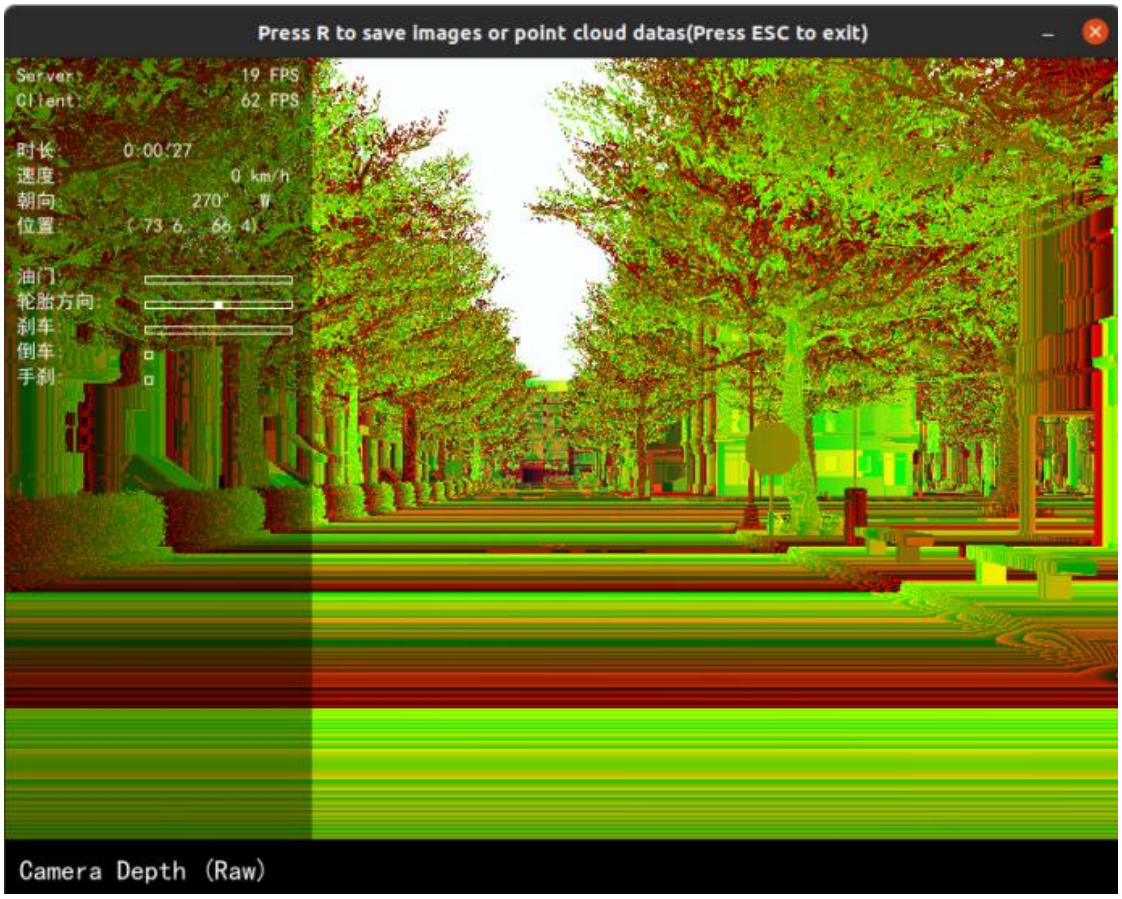

VisionVoyage Client
FILE BOX
Type here
Open
普通图像转换结果
普通图像拼接结果
分割结果
普通图像转鱼眼图像
五张普通图像拼接鱼眼图像
上传图像进行分割
上传视频进行实时分割
By: InquinityDrive
v6.5.0-stable


VisionVoyage Client
FILE BOX
Type here
Open
普通图像转换结果
普通图像拼接结果
分割结果
请输入手机号
请输入输入您支付宝绑定的手机号:
Cancel
OK
普通图像转鱼眼图像
五张普通图像拼接鱼眼图像
上传图像进行分割
上传视频进行实时分割
By: InquinityDrive
v6.5.0-stable

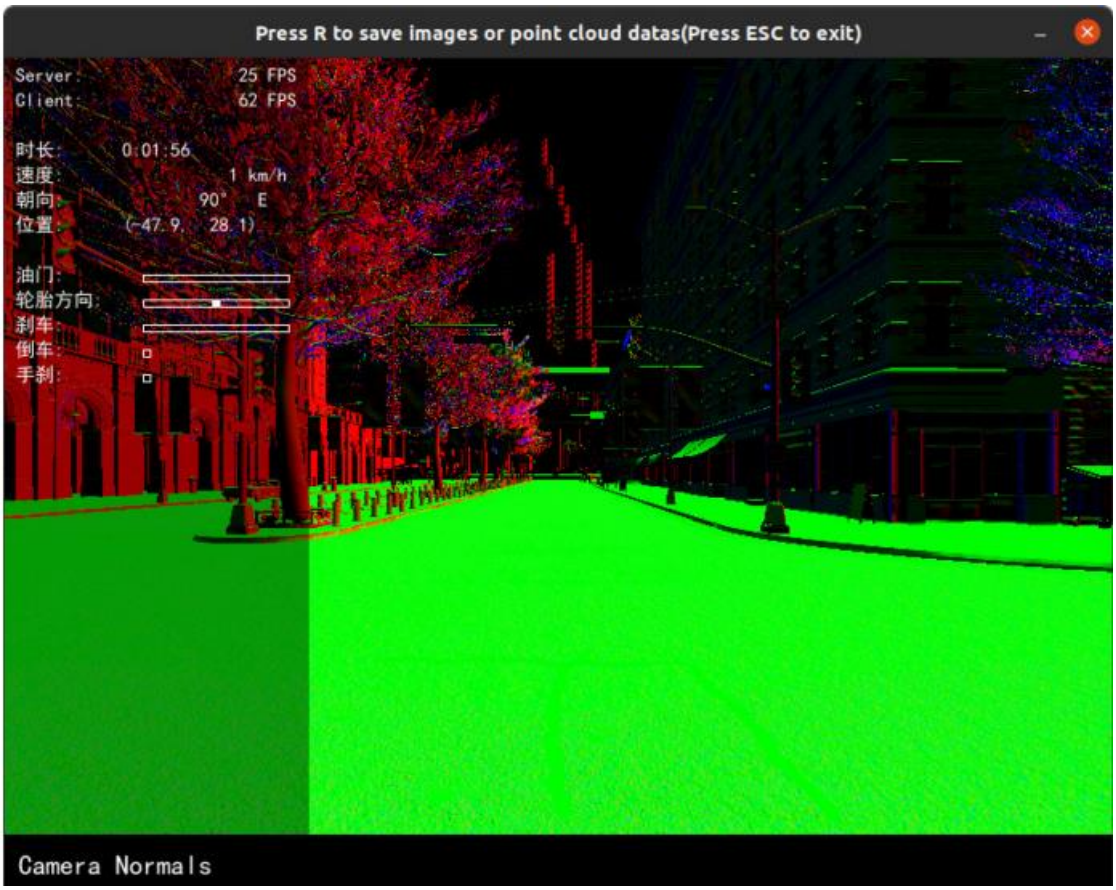


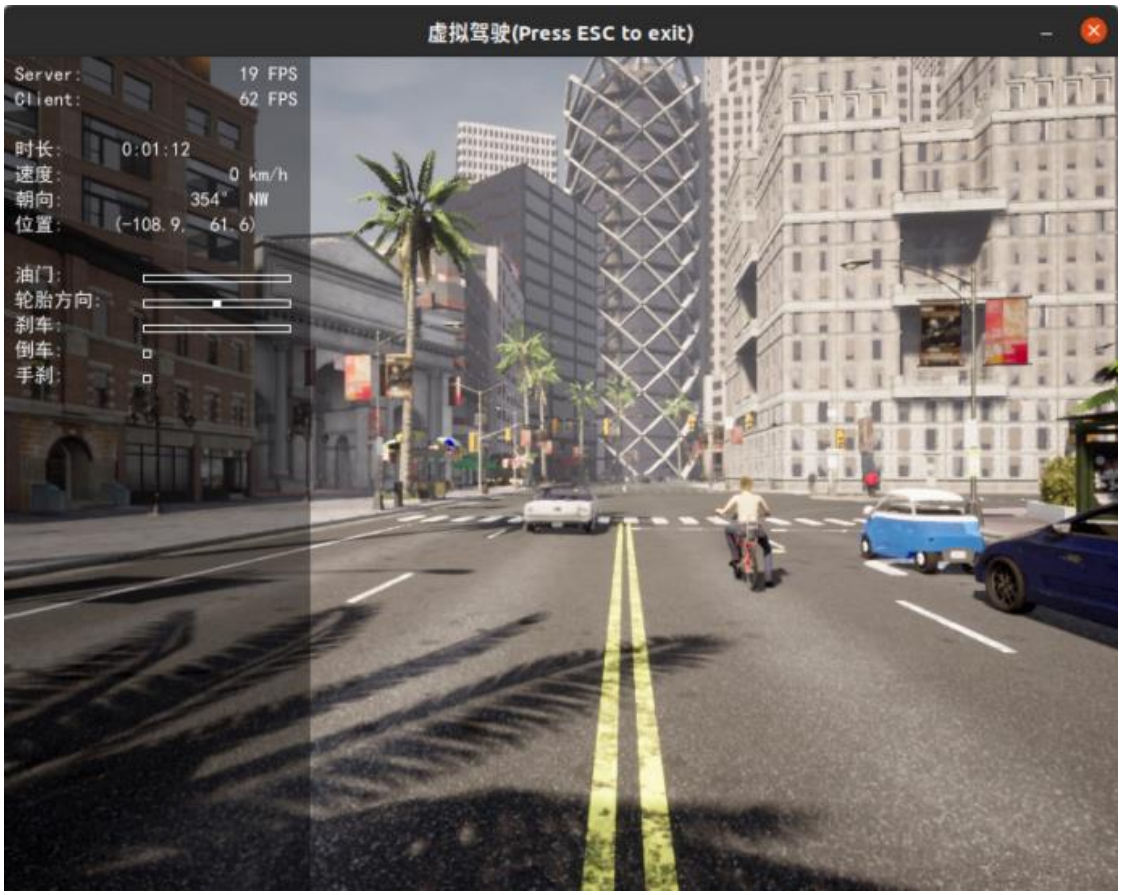

Press R to save images or point cloud datas(Press ESC to exit)
Server: 14 FPS
Client: 62 FPS
时长: 0:00:33
速度: 0 km/h
朝向: 270° W
位置: (73.6, 66.4)
油门:
轮胎方向:
刹车:
倒车:
手刹:
STOP
STOP
Camera RGB


Press R to save images or point cloud datas(Press ESC to exit)
Server: 19 FPS
Client: 62 FPS
时长: 0:00:47
速度: 0 km/h
朝向: 270° W
位置: (73.6, 66.4)
油门:
轮胎方向:
刹车:
倒车:
手刹:
Camera Semantic Segmentation (Raw)


Press R to save images or point cloud datas(Press ESC to exit)
Server: 19 FPS
Client: 62 FPS
时长: 0:00:27
速度: 0 km/h
朝向: 270° W
位置: (73.6, 66.4)
油门:
轮胎方向:
刹车:
倒车:
手刹:
Camera Depth (Raw)


Press R to save images or point cloud datas(Press ESC to exit)
Server: 22 FPS
Client: 62 FPS
时长: 0:00:53
速度: 0 km/h
朝向: 270° W
位置: (73.6, 66.4)
油门:
轮胎方向:
刹车:
倒车:
手刹:
STOP
Camera Semantic Segmentation (CityScapes Palette)


Press R to save images or point cloud datas(Press ESC to exit)
Server: 25 FPS
Client: 62 FPS
时长: 0:01:56
速度: 1 km/h
朝向: 90° E
位置: (-47.9, 28.1)
油门:
轮胎方向:
刹车:
倒车:
手刹:
Camera Normals


虚拟驾驶(Press ESC to exit)
Server: 19 FPS
Client: 62 FPS
时长: 0:01:12
速度: 0 km/h
朝向: 354° NW
位置: (-108.9, 61.6)
油门:
轮胎方向:
刹车:
倒车:
手刹:

个人遇到的主要问题及解决方法：

技术集成困难：

- **问题描述**：项目需要与多个外部系统和服务进行集成，但面临不同平台和协议的技术难题，导致集成进展缓慢。
- **解决方法**：采用了逐步集成和增量开发的策略，先进行基础架构的集成测试，确保系统间通信稳定，再逐步扩展到更复杂的功能。同时，与外部供应商和技术团队保持密切沟通，解决技术难题和协调接口。

资源限制与时间压力：

- **问题描述**：项目进度紧张，资源（人力、技术、时间）有限，难以满足高质量的开发和测试要求。
- **解决方法**：优先级管理和资源调配是关键。通过制定详细的项目计划和任务分解，合理分配开发和测试资源。同时，优化工作流程和提高团队效率，使用自动化测试和部署工具，减少重复工作，提升开发效率和质量。

团队沟通和协作挑战：

- **问题描述**：作为一个由五人组成的小组开发项目，我们面临内部沟通和协作效率低的问题，导致信息传递不畅和决策延迟。

- **解决方法：**为了应对这一挑战，我们致力于建立良好的内部沟通和协作机制。我们定期召开会议和沟通工作坊（每天早上开站会）明确项目目标、分配任务和确定责任。采用协作工和项目管理平，以促进信息共享、任务跟踪和团队协作。通过这些举措，我们确保团队成员之间的信息对称和资源协同使用，从而提升项目执行效率和问题解决能力。

个人实习体会及收获：

在最近的实习经历中，我获得了许多宝贵的经验和深刻的体会。以下是我在实习过程中的主要收获：

1. **实践机会与技能提升：**通过参与实际项目，我有机会将课堂学习应用到实际工作中。特别是在软件开发和团队协作方面，我学到了如何有效地使用版本控制工具、编写清晰的文档以及与团队成员合作解决技术问题。
2. **跨功能团队合作：**在跨功能团队中工作让我深刻体会到沟通和协作的重要性。通过与产品经理和开发人员的密切合作，我学会了如何在不同角色和背景的人员之间建立有效的工作关系，共同推动项目进展。
3. **问题解决与适应能力：**面对项目中的技术挑战和时间限制，我学会了快速分析问题、制定解决方案并迅速调整。这种经验不仅提升了我的解决问题的能力，还增强了我的适应能力和应对压力的能力。
4. **职业发展意识：**通过实习，我更清晰地了解了自己的职业兴趣和发展方向。我意识到在软件开发领域的深入学习和技能提升对未来职业发展至关重要，并计划在此基础上继续努力。
5. **自我管理与成长：**在实习过程中，我学会了有效地管理时间、设置优先级和自我激励。这些能力不仅帮助我在实习期间取得了积极成果，也为未来的职业生涯打下了坚实的基础。

通过这次实习，我不仅获得了宝贵的专业经验，还发展了自己的职业技能和领导能力，为将来的职业生涯奠定了坚实的基础。

生产实习成绩汇总

生产实习分数	平时成绩（20 分）		项目成绩（60 分）		报告成绩（20 分）		总分
	出勤情况 (10 分)	课堂表现 (10 分)	基本功能 (30 分)	答辩情况 (30 分)	报告内容 (10 分)	规范程度 (10 分)	

总体评价：

实习导师签名：

企业导师签名：