实践说明：Inverse perspective mapping

1. 实践目的：

通过实际的编程练习，让学生深入了解视图转换与IPM(Inverse perspective mapping) 原理，通过编写透视投影代码，掌握其核心数学原理。通过实际操作，学生将应用IPM将前视图映射到鸟瞰图，理解其在自动驾驶中的应用，培养问题解决和动手能力，加深对自动驾驶技术的理解。

2. 实践内容：

2.1 任务描述：

逆透视变换（IPM）是将相机视角转换成鸟瞰图的一种方法，下面我们进行IPM的原理说明：

涉及的三维坐标系：

- 世界坐标系 $(X_w, Y_w, Z_w)^T$
- 相机坐标系 $(X_c, Y_c, Z_c)^T$
- 默认相机坐标系 $(X_d, Y_d, Z_d)^T$
- 道路坐标系 $(X_r, Y_r, Z_r)^T$
- 道路坐标系ISO8855 $(X_i, Y_i, Z_i)^T$

默认相机坐标系的三条轴X，Y，Z分别指向车的右方，下方和前方，实际相机坐标系可能与默认相机坐标系有差别，本作业中的相机坐标系围绕x轴旋转（俯仰角pitch_deg=-5）

如图所示：

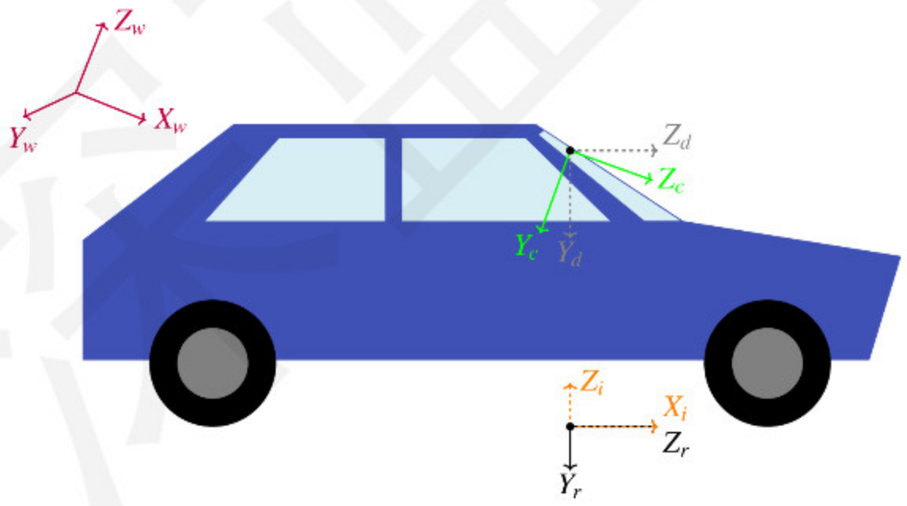


图1：图为侧视图，第3个坐标轴的方向可根据右手坐标系原则确定

三维坐标系之间的转换：

给定世界坐标系中一点$(X_w, Y_w, Z_w)^T$，可通过旋转矩阵$\mathbf{R} \in \mathbb{R}^{3\times3}$和平移向量$\mathbf{t}$，转换到相机坐标系下，公式如下：

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = \mathbf{R} \begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} + \mathbf{t}$$

将点采用齐次坐标来表示，变换如下：

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix} = \begin{pmatrix} R_{xx} & R_{xy} & R_{xz} & t_x \\ R_{yx} & R_{yy} & R_{yz} & t_y \\ R_{zx} & R_{zy} & R_{zz} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} = \mathbf{T}_{cw} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

其中，$\mathbf{T_{cw}}$被称为变换矩阵（transformation matrix），任意两个三维坐标系之间的转换都可以用上述方式。

三维坐标系和二维坐标系的转换：

给定一个点P，它在相机坐标系下的坐标为 $(X_c, Y_c, Z_c)^T$，通过下述公式可以得到该点的像素坐标(u, v)：

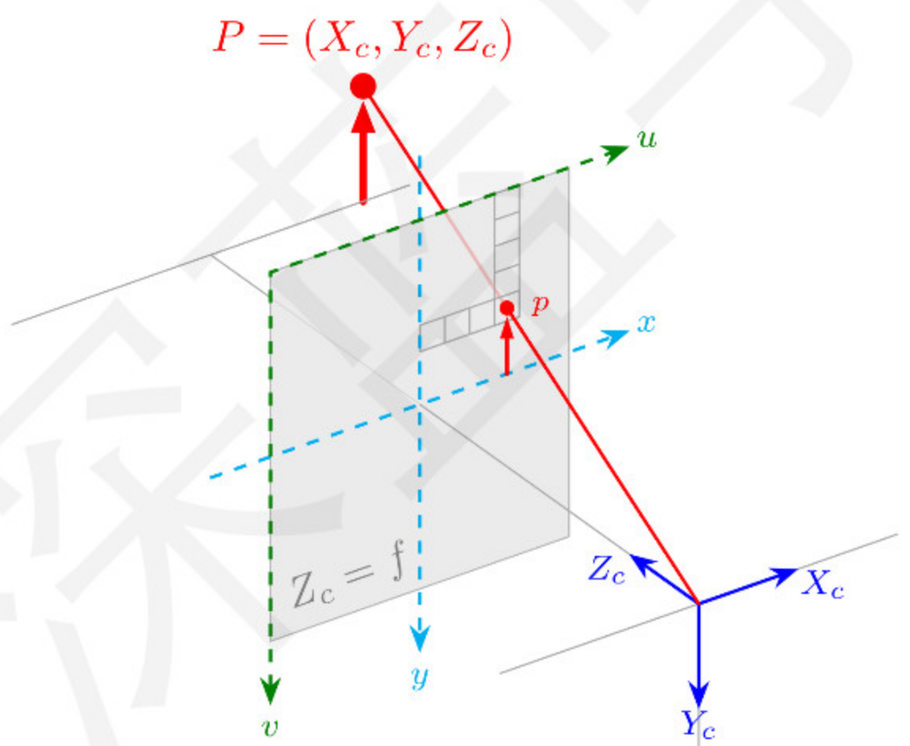
$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \mathbf{K} \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix}$$

其中，$\mathbf{K}$表示相机内参矩阵（camera instrinsic），λ表示三维点P在相机坐标系下的深度。

现在给定像素坐标(u, v)，需要找到该点对应的相机坐标系下的坐标 $(X_c, Y_c, Z_c)^T$，只需要将上述公式两边乘上$\mathbf{K}^{-1}$，即：

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = \lambda \mathbf{K}^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$$

如图所示，像素坐标(u, v)对应的相机坐标系下的3D点 $P(X_c, Y_c, Z_c)^T$在一条直线上

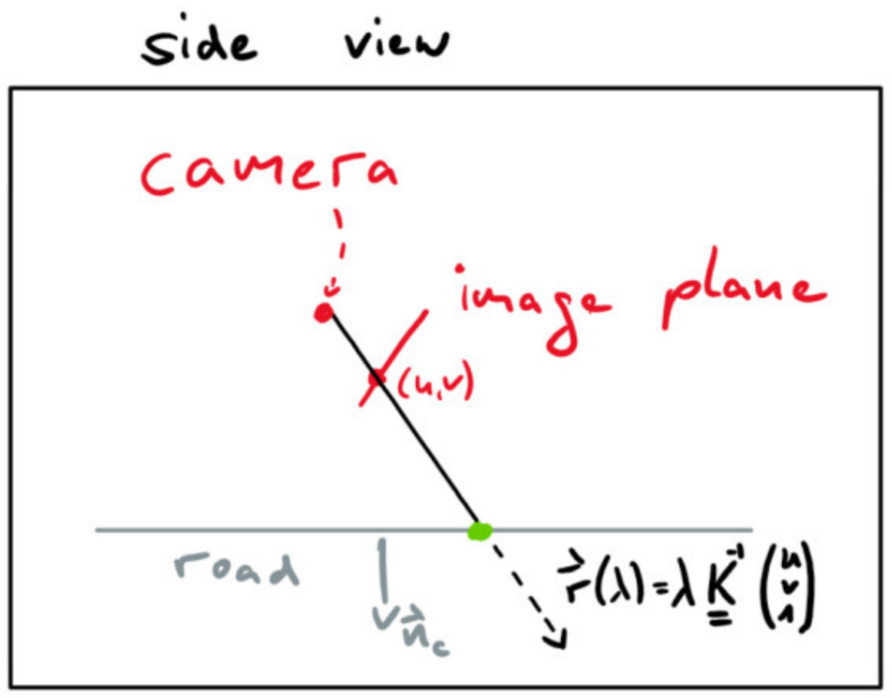


所以点P可定义成关于λ的函数：

$$\mathbf{r}(\lambda) = \lambda \mathbf{K}^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}, \ \lambda \in \mathbb{R}_{>0} \tag{1}$$

然而问题是，我们无法确定λ的值。IPM方法假定$\mathbf{r}(\lambda)$应该位于路面上，即假设道路是平坦的，将像素投影到地平面。

如图所示：（图为侧视图）



根据法向量$\mathbf{n}$和平面上的点$\mathbf{r}_0$，给出道路的平面方程：
$\mathbf{n}^T(\mathbf{r}-\mathbf{r}_0)=0$
在道路坐标系下，法向量为 $\mathbf{n}=(0,1,0)^T$。

由于相机的光轴与地面不平行，法向量$\mathbf{n}$在相机坐标系下表示为 $\mathbf{n_c}=\mathbf{R_{rc}}(0,1,0)^T$，$\mathbf{R_{rc}}$ 为旋转矩阵，可将道路参考系下的向量转换到相机参考系下。

在相机参考系下，相机中心位于坐标 $(0,0,0)^T$，设h为相机相对路面的高度，沿着路面法向量$\mathbf{n_c}$ 方向，将点 $(0,0,0)^T$移动距离h，可得到路平面上一点，故令$\mathbf{r}_0=h\mathbf{n_c}$。

将$\mathbf{r}_0=h\mathbf{n_c}$代入上述平面方程，得

$0=\mathbf{n_c}^T(\mathbf{r}-\mathbf{r}_0)=\mathbf{n_c}^T\mathbf{r}-h$
或 $h=\mathbf{n_c}^T\mathbf{r}$
将公式(1)，即 $\mathbf{r}(\lambda)=\lambda\mathbf{K}^{-1}(u,v,1)^T$，带入平面方程$h=\mathbf{n_c}^T\mathbf{r}$，可得：

$$h=\mathbf{n_c}^T\lambda\mathbf{K}^{-1}(u,v,1)^T \Leftrightarrow \lambda=\frac{h}{\mathbf{n_c}^T\mathbf{K}^{-1}(u,v,1)^T}$$

求出 λ 后，将其代入$\mathbf{r}(\lambda)$即可求出相机坐标系下的三维坐标$(X_c,Y_c,Z_c)^T$：

$$\begin{pmatrix}X_c\\Y_c\\Z_c\end{pmatrix}=\frac{h}{\mathbf{n_c}^T\mathbf{K}^{-1}(u,v,1)^T}\mathbf{K}^{-1}\begin{pmatrix}u\\v\\1\end{pmatrix} \tag{2}$$

2.2 任务步骤：

2.2.1 环境准备

```
pip install -r requirements.txt
```

2.2.2 代码实践

找到**TODO**所在位置：`exercises/camera_geometry.py`，补全代码

实现说明：

TODO: 补全`__init__`中部分变量的初始化

```
```

```
    def __init__(self, height=1.3, yaw_deg=0, pitch_deg=-5, roll_deg=0,
image_width=1024, image_height=512, field_of_view_deg=45):
        # scalar constants
        self.height = height
        self.pitch_deg = pitch_deg
        self.roll_deg = roll_deg
        self.yaw_deg = yaw_deg
        self.image_width = image_width
        self.image_height = image_height
        self.field_of_view_deg = field_of_view_deg
        # camera intriniscs and extrinsics
        self.intrinsic_matrix = get_intrinsic_matrix(field_of_view_deg, image_width,
image_height)
        self.inverse_intrinsic_matrix = np.linalg.inv(self.intrinsic_matrix)
        ## Note that "rotation_cam_to_road" has the math symbol R_{rc} in the book
        yaw = np.deg2rad(yaw_deg)
        pitch = np.deg2rad(pitch_deg)
        roll = np.deg2rad(roll_deg)
        cy, sy = np.cos(yaw), np.sin(yaw)
        cp, sp = np.cos(pitch), np.sin(pitch)
        cr, sr = np.cos(roll), np.sin(roll)
        rotation_road_to_cam = np.array([[cr*cy+sp*sr+sy, cr*sp*sy-cy*sr, -cp*sy],
                                         [cp*sr, cp*cr, sp],
                                         [cr*sy-cy*sp*sr, -cr*cy*sp -sr*sy,
cp*cy]])
        self.rotation_cam_to_road = rotation_road_to_cam.T # for rotation matrices,
taking the transpose is the same as inversion

        # TODO replace the 'None' values in the following code with correct
expressions

        self.translation_cam_to_road = None
        self.trafo_cam_to_road = None
        # compute vector nc. Note that R_{rc}^T = R_{cr}
        self.road_normal_camframe = None
```

提示：

- self.translation_cam_to_road 表示相机坐标系到路面坐标系的平移向量$\mathbf{t}$，坐标系方向参考图1；
- self.trafo_cam_to_road 表示相机坐标系到路面坐标系的变换矩阵$T_{cr} \in \mathbb{R}^{4\times4}$；
- self.road_normal_camframe 表示相机参考系下的路平面法向量$\mathbf{n_c} = \mathbf{R_{rc}}(0,1,0)^T$，原理参考2.1节的内容。

TODO: 根据公式(2)实现逆透视变换，将像素坐标(u,v)转换成对应的相机坐标系下的三维坐标$(X_c, Y_c, Z_c)^T$

```
    def uv_to_roadXYZ_camframe(self,u,v):
        """
        Inverse perspective mapping from pixel coordinates to 3d coordinates.

        Parameters
```

```
        ----------
        u,v: Both float
            Pixel coordinates of some part of the road.

        Returns:
        --------
        XYZ: array_like, shape(3,)
            Three dimensional point in the camera reference frame that lies on the
road
            and was mapped by the camera to pixel coordinates u,v
        """
        # TODO  Write this function
        raise NotImplementedError
```

TODO: 将坐标从相机坐标系转换到道路坐标系

```
    def camframe_to_roadframe(self,vec_in_cam_frame):
        """
        Transform coordinates from camera reference frame to road reference frame.

        Parameters
        ----------
        vec_in_cam_frame: array_like, shape(3,)
            Three dimensional point in the camera reference frame that lies on the
road

        Returns:
        --------
        XYZ: array_like, shape(3,)
            Three dimensional point in the road reference frame that lies on the
road
        """
        # TODO: Write this function
        raise NotImplementedError
```

提示：参考公式

$$\begin{pmatrix} X_r \\ Y_r \\ Z_r \end{pmatrix} = \mathbf{R} \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} + \mathbf{t}$$

TODO: 将像素坐标(u, v)转换成对应的iso8855道路坐标系下的坐标 $(X_i, Y_i, Z_i)^T$

```
    def uv_to_roadXYZ_roadframe_iso8855(self,u,v):
        """
        Inverse perspective mapping from pixel coordinates to 3d coordinates in road
frame iso8855.

        Parameters
        ----------
        u,v: Both float
            Pixel coordinates of some part of the road.
```

```
        Returns:
        --------
        XYZ: array_like, shape(3,)
            Three dimensional point in the road reference frame iso8855 that lies on
the road
            and was mapped by the camera to pixel coordinates u,v
        """
        # TODO: Write this function
        raise NotImplementedError
```

提示：坐标系方向参考图1，注意区分道路坐标系和道路坐标系ISO8855

2.2.3 测试

找到测试脚本`code/tests/inverse_perspective_mapping.ipynb`，依次执行cells，通过可视化验证你的实现是否正确。

3. 提交格式：

- pdf文件：`report.pdf`，包含**2.2.3**小节中生成的4张图片：raw_image.jpg、boundary_world_frame.jpg、boundary_road_frame_1.jpg、boundary_road_frame_2.jpg；
- `camera_geometry.py`文件。

将上述文件一起打包成zip，以自己的深蓝用户名命名。