**世界坐标到像素坐标投影系统说明文档**

**概述**

本系统实现了一个完整的3D世界坐标（以足球场中央为坐标原点，图中的禁区角点的世界坐标为(7,-3,0)）到2D像素坐标的投影工具，能够将三维空间中的点投影到相机图像平面，并在图像上可视化标记这些点。系统包含相机建模、坐标变换、畸变校正和可视化等功能。

**核心类说明**

**Point3D**

**功能**: 表示三维空间中的点

python

class Point3D:

def \_\_init\_\_(self, x=0.0, y=0.0, z=0.0):

self.x = x *# X坐标*

self.y = y *# Y坐标*

self.z = z *# Z坐标*

**Point2D**

**功能**: 表示二维图像平面中的点

python

class Point2D:

def \_\_init\_\_(self, x=0.0, y=0.0):

self.x = x *# X坐标*

self.y = y *# Y坐标*

**CameraIntrinsics**

**功能**: 相机内参模型，包含焦距、主点和畸变参数

python

class CameraIntrinsics:

def \_\_init\_\_(self, fx=500.0, fy=500.0, cx=640.0, cy=360.0,

k1=0.0, k2=0.0, k3=0.0, p1=0.0, p2=0.0):

self.fx = fx *# X轴焦距*

self.fy = fy *# Y轴焦距*

self.cx = cx *# 主点X坐标*

self.cy = cy *# 主点Y坐标*

self.k1 = k1 *# 径向畸变系数1*

self.k2 = k2 *# 径向畸变系数2*

self.k3 = k3 *# 径向畸变系数3*

self.p1 = p1 *# 切向畸变系数1*

self.p2 = p2 *# 切向畸变系数2*

**TransformMatrix**

**功能**: 4×4齐次变换矩阵，用于表示三维空间中的刚体变换

python

class TransformMatrix:

def \_\_init\_\_(self):

self.data = np.eye(4, dtype=np.float64) *# 4×4单位矩阵*

**BackProjector**

**功能**: 核心投影器，实现世界坐标到像素坐标的完整投影流程

**主要方法:**

* world\_to\_camera(): 世界坐标 → 相机坐标
* camera\_to\_image(): 相机坐标 → 理想图像坐标
* apply\_distortion(): 应用镜头畸变校正
* image\_to\_pixel(): 图像坐标 → 像素坐标
* world\_to\_pixel(): 完整投影流程（世界坐标 → 像素坐标）

**WorldPointMarker**

**功能**: 在图像上可视化标记世界坐标点

python

def mark\_world\_point(image, world\_point, back\_projector, label="")

* 将世界坐标投影到像素坐标
* 在对应位置绘制心形标记
* 添加坐标文本标签

**坐标变换流程**

**1. 世界坐标 → 相机坐标**

使用外参矩阵进行刚体变换：

text

[camera\_point] = [R|t] × [world\_point]

**2. 相机坐标 → 理想图像坐标**

透视投影模型：

text

x\_ideal = (X\_camera / Z\_camera) × fx

y\_ideal = (Y\_camera / Z\_camera) × fy

**3. 应用镜头畸变**

使用Brown-Conrady畸变模型：

text

径向畸变: (1 + k1·r² + k2·r⁴ + k3·r⁶)

切向畸变: 2p1·xy + p2·(r²+2x²), p1·(r²+2y²) + 2p2·xy

**4. 图像坐标 → 像素坐标**

坐标平移：

text

x\_pixel = x\_distorted + cx

y\_pixel = y\_distorted + cy

**工具函数**

**创建旋转矩阵**

* create\_rotation\_x(angle\_degrees): 绕X轴旋转
* create\_rotation\_y(angle\_degrees): 绕Y轴旋转
* create\_rotation\_z(angle\_degrees): 绕Z轴旋转

**创建平移矩阵**

* create\_translation(tx, ty, tz): 创建平移变换矩阵

**创建外参矩阵**

* create\_extrinsic\_matrix(pos\_x, pos\_y, pos\_z, pitch\_deg, yaw\_deg, roll\_deg)
  + 根据相机位置和欧拉角创建完整的外参矩阵

**使用示例**

**基本用法**

python

*# 1. 设置相机参数*

intrinsics = CameraIntrinsics(fx=645.06, fy=644.26, cx=649.56, cy=373.50)

*# 2. 创建外参矩阵*

extrinsic = create\_extrinsic\_matrix(

pos\_x=6.0, pos\_y=-4.5, pos\_z=1.0, *# 相机位置*

pitch\_deg=0.0, yaw\_deg=-15.0, roll\_deg=-115.0 *# 相机姿态*

)

*# 3. 创建投影器*

back\_projector = BackProjector(intrinsics, extrinsic)

*# 4. 创建标记器*

marker = WorldPointMarker()

*# 5. 投影并标记世界点*

world\_point = Point3D(7, -3, 0)

result\_image = marker.mark\_world\_point(image, world\_point, back\_projector, "Target")

**测试函数**

python

test\_world\_point\_projection("football\_image.jpg")

**输入输出**

**输入要求**

* **图像文件**: 支持常见图像格式（JPG、PNG等）
* **相机内参**: 焦距、主点坐标、畸变系数
* **相机外参**: 位置坐标和欧拉角
* **世界坐标**: 需要投影的三维点坐标

**输出结果**

* **控制台输出**: 详细的投影过程信息
* **可视化图像**: 带有标记点的结果图像
* **保存文件**: 自动保存为"world\_point\_projection\_result.jpg"

**注意事项**

1. **坐标系约定**:
   * 世界坐标系: 右手坐标系
   * 相机坐标系: Z轴向前，Y轴向下，X轴向右
   * 图像坐标系: 原点在图像中心
2. **投影限制**:
   * 仅投影在相机前方的点（Z\_camera > 0）
   * 不支持点在无穷远的情况
3. **畸变模型**: 使用标准的Brown-Conrady畸变模型，支持径向和切向畸变
4. **数值稳定性**: 包含除零保护和数值容错处理

**应用场景**

* 计算机视觉中的3D-2D投影
* 相机标定验证
* 增强现实应用
* 机器人视觉系统
* 运动分析中的坐标转换

这个系统提供了一个完整、可扩展的框架，用于处理三维空间点到二维图像平面的投影问题，特别适合需要精确相机建模和可视化验证的应用场景。