# Computer Vision homework 3

Pan Changxun

April 2025

## 1 3D Location Transformation

a) The intrinsic matrix is given by:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

When $f_x = f_y = f = 721.5$ and $c_x, c_y = (609.6, 172.9)$, the intrinsic matrix becomes:

$$K = \begin{bmatrix} 721.5 & 0 & 609.6 \\ 0 & 721.5 & 172.9 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

b) We use the camera's coordinates: $x$ means the right, $y$ means the down, $z$ means the forward. Then the equation of the ground plane in camera's coordinate system is $y = 1.7m$.

c) We use the same camera's coordinate system as in part b). When given a 2D point (x,y) in the image and suppose it lies on the ground (y=1.7m), we can find the corresponding 3D point in the camera's coordinate system using the following equations:

$$Y = 1.7m \tag{3}$$

$$Z = \frac{Y \cdot f_y}{y - c_y} = \frac{1.7m \cdot 721.5}{y - 172.9} \tag{4}$$

$$X = Y \cdot \frac{x - c_x}{y - c_y} = 1.7m \cdot \frac{x - 609.6}{y - 172.9} \tag{5}$$

where (X, Y, Z) are the coordinates of the corresponding 3D point in the camera's coordinate system and $c_x, c_y$ equals to (609.6, 172.9).

## 2 Road Analyzing

a) For the depth map visualization, I truncated the 5% most distant pixels to enhance the details of nearer objects. This approach provides better contrast and clarity in the depth representation. The resulting depth maps are stored in the folder "/src/problem2_analyzing/data/depth/" with corresponding colorbars. An example depth map is shown below:

Figure 1: Depth map visualization for frame 004945

The corresponding colorbar for interpreting the depth values:

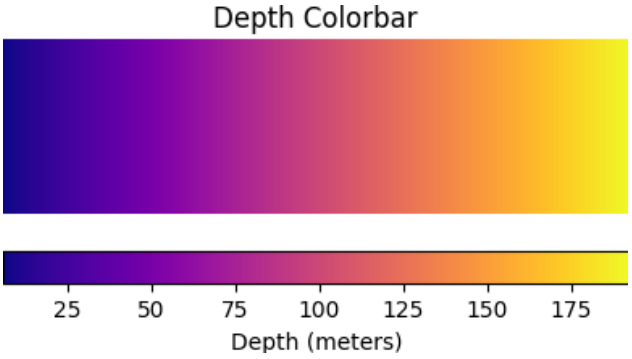


Figure 2: Colorbar for depth map 004945

b) The bounding box visualizations are stored in "/src/problem2_analyzing/data/bbox/". A representative example is shown here:

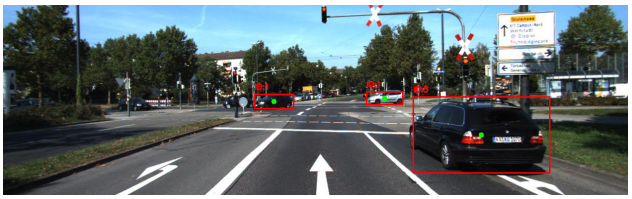

Figure 3: Bounding box visualization for frame 004945

The calculated 3D coordinates for the centers of the bounding boxes are stored in "/src/problem2_analyzing/data/3d_c
Selected examples from frame 004945:

| ID | 2D coordinates | 3D coordinates (meters) |
|---|---|---|
| 3 | [950, 256] | [3.239, 0.791, 6.864] |
| 1 | [758, 184] | [9.885, 0.742, 48.046] |
| 1 | [538, 190] | [-3.465, 0.830, 34.943] |

# 3 Self-driving Detection (Bonus)

The results are stored in "/src/problem3_driving/output/".
Here are the keypoints of the self-driving detection algorithm:

```python
# nonplayer local rotate
nonplayer_rotate_tran = BoundingBoxesTransform._get_nonplayer_rotate_transform(nonplayer_transform)
np_rotate = nonplayer_rotate_tran @ bb_cords.T  # 4x8矩阵

# nonplayer to player
nonplayer_player_tran = BoundingBoxesTransform._get_player_transform(nonplayer_transform, player_transform, z_box_local_nonplayer, z_box_local_player)
player_unrotate = nonplayer_player_tran @ np_rotate  # 4x8矩阵

# player rotate
player_rotate_tran = BoundingBoxesTransform._get_player_rotate_matrix(player_transform)
player_rotate = player_rotate_tran @ player_unrotate  # 4x8矩阵

# player to camera
camera_tran = BoundingBoxesTransform._get_camera_matrix(camera_transform)
camera_unrotate = camera_tran @ player_rotate  # 4x8矩阵

# camera rotate
camera_rotate_tran = BoundingBoxesTransform._get_view_matrix(camera_transform)
camera_view = camera_rotate_tran @ camera_unrotate  # 4x8矩阵
```

Figure 4:

```python
# Apply camera calibration matrix
points_2d = camera.calibration @ points_2d
points_2d = points_2d[:2, :]  / points_2d[2, :]
points_2d = points_2d[:2, :]

# Flip Y-axis to match image coordinates (origin at top-left)
points_2d[1, :] = IMAGE_HEIGHT - points_2d[1, :]
```

Figure 5:

Here are the results of the self-driving detection algorithm: (3 figures)

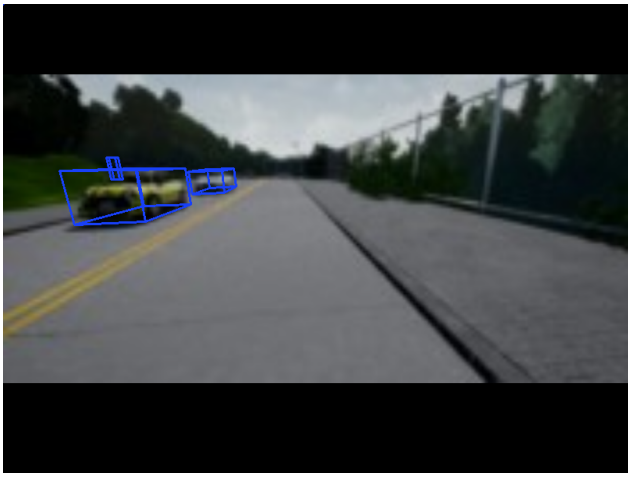

Figure 6:

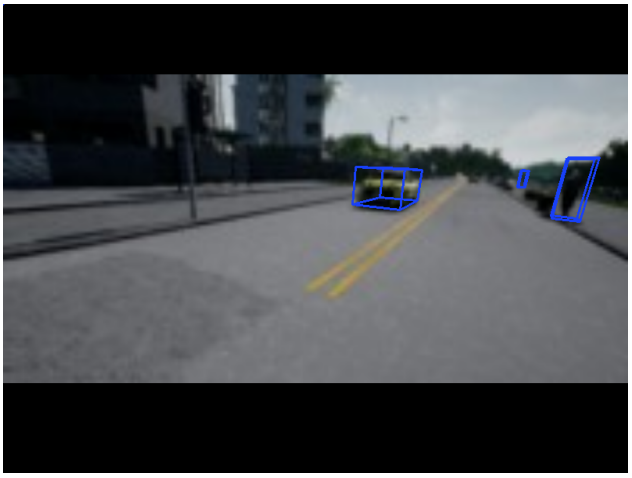

Figure 7:

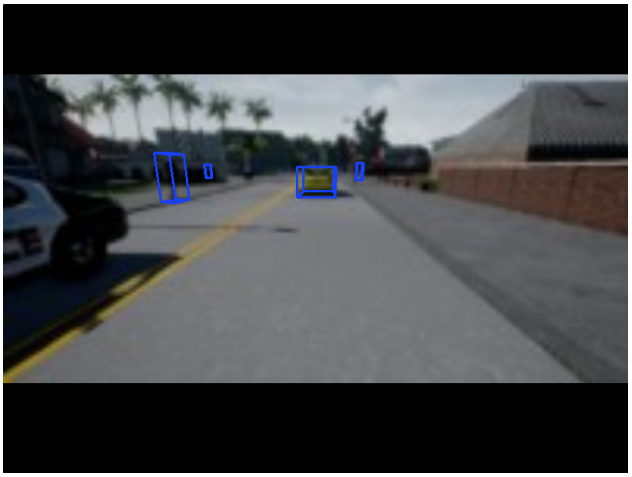

Figure 8: