1 Theory

1.1 Pinhole camera model

The most common imaging model used in computer vision is the pinhole camera model[1], which is illustrated in Fig. 1.

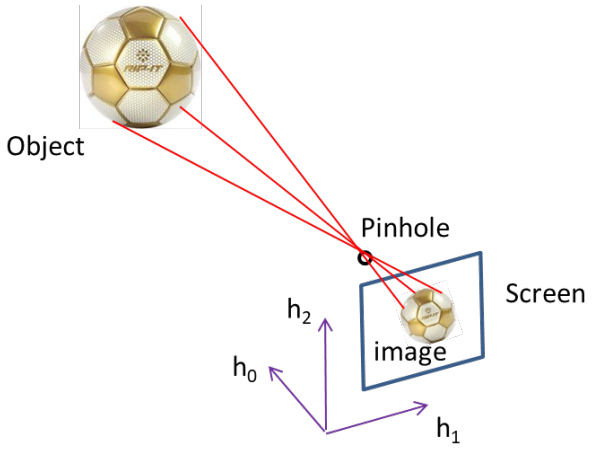

Figure 1: Pinhole camera model

Without loss of generality, suppose the pinhole is located at $(0, 0, 0)$. Denote the position of an object by $\vec{r} = (x, y, z)$, its velocity by $\vec{v} = (v_x, v_y, v_z)$, and its acceleration by $\vec{a} = (a_x, a_y, a_z)$. The screen on which the image is captured is located at a vector $\vec{h_s} = h_s\hat{h_0}$ from the origin. The screen coordinates basic vectors are $\hat{h_0}$, $\hat{h}_1$ and $\hat{h}_2$. That is, $< \hat{h_0}, \hat{h_1}, \hat{h_2} >$ also form a right-hand coordinate system. Note that

$$\hat{h_0} \perp \hat{h_1} \perp \hat{h_2} \tag{1}$$

$$|\hat{h_0}| = |\hat{h_1}| = |\hat{h_2}| = 1 \tag{2}$$

The screen plane is given by

$$h_s\hat{h_0} + a\hat{h_1} + b\hat{h_2}, \quad a, b \in R \tag{3}$$

Then the projection of $\vec{r}$ on the screen, which is the image of the object, $\vec{p} = (a, b)$, is given by

$$-k\vec{r} = h_s\hat{h_0} + a\vec{h_1} + b\vec{h_2} \tag{4}$$

where k is the distance of the object from the pinhole and a and b are the "screen coordinates" of its image.

$$\left(\vec{r}, \hat{h_1}, \hat{h_2}\right) \begin{pmatrix} k \\ a \\ b \end{pmatrix} = -h_s\hat{h_0} \tag{5}$$

[1]David A. Forsyth and Jean Ponce (2003). Computer Vision, A Modern Approach. Prentice Hall. ISBN 0-12-379777-2.

or

$$\left(\hat{h_1}, \hat{h_2}, \vec{r}\right) \begin{pmatrix} a \\ b \\ k \end{pmatrix} = -h_s \hat{h_0}$$

Define

$$C^{-1} = \left(\hat{h_1}, \hat{h_2}, \vec{r}\right)^{-1} \tag{6}$$

as the inverse camera matrix. The screen coordinates of the object (a, b), as well as its distance k from the pinhole, are given by

$$\underline{\begin{pmatrix} a \\ b \\ k \end{pmatrix} = -C^{-1} h_s \hat{h_0}} \tag{7}$$

1.1.1 The characteristics of the screen coordinate system

The world coordinates are denoted by $(\hat{x}, \hat{y}, \hat{z})$, where $\hat{z}$ is the upward direction and $(\hat{x}, \hat{y})$ spans the ground. In the real world situations that we hope to analyze, not only do the objects move, but the camera also moves simultaneously. Therefore, $< \hat{h_0}, \hat{h_1}, \hat{h_2} >$ translates and rotates around the pinhole, and the pinhole also moves. In typical scenarios, the camera stands vertically, so $\vec{h_2}$ is close to vertical, the angle between them being θ. Then

$$\hat{h_2} \cdot \hat{z} = \cos\theta \approx 1 \tag{8}$$

1.1.2 The solution to the imaging equation

In Eq. (4)

$$k\vec{r} + h_s \hat{h_0} + a\vec{h_1} + b\vec{h_2} = 0$$

Multiplying the equation by $\hat{h_i}$ for $i = 0, 1, 2$, noting that

$$\hat{h}_i \cdot \hat{h}_j = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}$$

We get

$$k\left(\vec{r} \cdot h\right) = -h_s$$
$$k\left(\vec{r} \cdot \hat{h_1}\right) + a = 0$$
$$k\left(\vec{r} \cdot h_2\right) + b = 0$$

Therefore

$$\underline{k = -\frac{h_s}{\vec{r} \cdot h}}, \quad a = -k\left(\vec{r} \cdot h\right) = \underline{h_s \frac{\vec{r} \cdot \hat{h_1}}{\vec{r} \cdot h}}, \quad b = \underline{h_s \frac{\vec{r} \cdot \hat{h_2}}{\vec{r} \cdot \hat{h_0}}} \tag{9}$$

The image is at (a, b) on the capturing device, typically a CCD or CMOS image sensor. Note that the final image shown in an video will be translated and zoomed according to the frame configuration. The final location of the pixel is

$$\left(a', b'\right) = (c_x, c_y) + M\left(a, b\right) \tag{10}$$

where (c_x, c_y) are the offset of the image sensor center to the video center and M is the zoom factor.

Note that if $\vec{r} \perp \hat{h_0}$,this object is out of the FOV (field-of-view) and not on the screen.

Similarly, if $\vec{r} \cdot \hat{h}_0 < 0$, the object is "behind" the camera and cannot form an image.

1.2 The bounding box of an object

Assuming the object's diameter is D, the whole shape can be approximated by a box whose vertices are given by

$$\vec{r_i} = \vec{r} + \left(\pm\frac{D}{2}, \pm\frac{D}{2}, \pm\frac{D}{2} \right) \tag{11}$$

where $1 \leq i \leq 8$. Then for each $\vec{r}_i$, its projection on the camera screen is $\vec{p}_i = (a'_i, b'_i)$, also obtained by Eq. (9).

Thus, the bounding box of the object's image is the range of the 8 projections:

$$ll_x = \min(a'_i), \quad ll_y = \min(b'_i), \quad 1 \leq i \leq 8 \tag{12}$$

$$ur_x = \max(a'_i), \quad ur_y = \max(b'_i), \quad 1 \leq i \leq 8 \tag{13}$$

where ll, ur denote the lower-left and upper-right corner, respectively. For the sake of simplicity, denote the bounding box of the object's image by

$$\vec{B} = \begin{pmatrix} ll_x \\ ll_y \\ ur_x \\ ur_y \end{pmatrix} = \begin{pmatrix} \min a'_i \\ \min b'_i \\ \max a'_i \\ \max b'_i \end{pmatrix} \tag{14}$$

1.3 Motion model

Denote the state of an object at the time t by

$$\vec{s}(t) = \begin{pmatrix} \vec{r} \\ \vec{v} \\ \vec{a} \end{pmatrix} \tag{15}$$

Then after a short time Δt, its state becomes

$$\vec{s}(t + \Delta t) = \begin{pmatrix} \vec{r} + \vec{v}\Delta t + \frac{1}{2}\vec{a}\,(\Delta t)^2 \\ \vec{v} + \vec{a}\Delta t \\ \vec{a} + \Delta\vec{a} \end{pmatrix} \tag{16}$$

The bounding box of its image changes to

$$\vec{B}\,(t) = f\,(\vec{r}) \rightarrow \vec{B}\,(t + \Delta t) = f\left(\vec{r} + \vec{v}\Delta t + \frac{1}{2}\vec{a}\,(\Delta t)^2 \right) \tag{17}$$

where $f\,(.)$ is the operator that maps the 3D location of an object to the bounding box of its image, described by Eq. (9) - (14).

2 Trajectory prediction algorithm

Given a sequence of image frames captured by a video camera, how can we predict the trajectory of an object. As human or animals do, we first need to locate the object in these frames and then predict its next location based on its current trajectory. And very often we need to use experiences about how such objects typically move.

YOLO (You Only Look Once) is a real-time object detection and image segmentation deep neural network (DNN) model, developed by Joseph Redmon and Ali Farhadi at the University of Washington[2]. Since its launching in 2015, YOLO quickly gained popularity for its high speed and accuracy. The latest model version is YOLO11.[3] The pre-trained model can detect 80 classes, among which is "sports ball" (class 32). In this project, I use YOLO11s to locate the bounding box of a soccer ball, shown in Fig 2.

Figure 2: The red box is the bounding box of the soccer ball

2.1 Assumptions

As a first-order approximation, I assume the video camera does not move, and it is perfectly aligned with the world coordinate system. That is

$$\left(\hat{h}_0, \hat{h}_1, \hat{h}_2\right) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

where the unit is meter. The distance of the camera screen to the optimal center (pinhole) is $h_s = 0.2$ meter. And the camera offset and zoom in Eq. (10) are taken as

$$(c_x, c_y) = (0, 0) \quad M = 1000$$

[2]https://docs.ultralytics.com/

[3]https://huggingface.co/Ultralytics/YOLO11

These numbers are rather arbitrary, but as shown later, they have little effects on prediction accuracy.

2.2 Algorithm

Given $n+1$ bounding boxes in the corresponding consecutive images, how to predict the bounding boxes in the next m image frames? The imaging process is illustrated in Fig. 3.

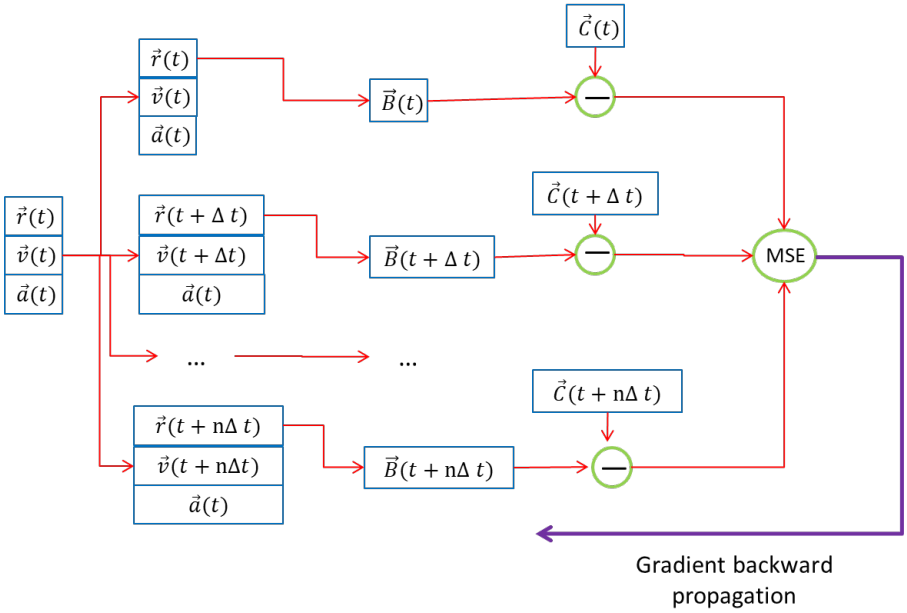

Figure 3: The equivalent neural network of video process