```
# Launch all of the node
python experiments/launch_nodes.py --robot=panda

# run the enviroment loop
python experiments/run_env.py --agent=gello
```

`run_env.py` 文件：

修改1：

```
from gello.agents.agent import BimanualAgent, DummyAgent
from gello.agents.gello_agent import GelloAgent
from gello.data_utils.format_obs import save_frame
from gello.env import RobotEnv
from gello.robots.robot import PrintRobot
from gello.zmq_core.robot_node import ZMQClientRobot

import os
import cv2
from ZEDCamera import ZedCamera

def print_color(*args, color=None, attrs=(), **kwargs):
    import termcolor
```

在同一个 `experiments` 文件夹下导入 `ZEDCamera.py` 文件，使得 Python 解释器可以解析：

```
from ZEDCamera import ZedCamera
```

修改2：

```
    print_color("\nStart 🚀🚀🚀", color="green", attrs=("bold",))

    zed_camera = ZedCamera(fps=30)
    zed_wrist_camera = ZedCamera(fps=30)

    save_path = None
    start_time = time.time()
```

初始化两个相机的接口，一个是主视角相机，一个是腕部视角相机

修改3：

```
266         elif state == "save":
267             assert save_path is not None, "something went wrong"
268 
269             # 在save_frame 之前获取图像，尽可能减少延迟
270             # 左边的图像，右边的图像，获取图像可能产生延迟
271             left_image, right_image = zed_camera.capture_frame()
272             wrist_left_image, wrist_right_image = zed_wrist_camera.capture_frame()
273 
274             save_frame(save_path, dt, obs, action)
275 
276             image_path = os.path.join(save_path, "image")
277             wrist_image_path = os.path.join(save_path, "wrist_image")
278 
279             print(f"image_path: {image_path}")
280             print(f"wrist_image_path: {wrist_image_path}")
281 
282             os.mkdir(image_path)
283             os.mkdir(wrist_image_path)
284 
285             dt = datetime.datetime.now()
286             timestamp = dt.strftime("%Y%m%d_%H%M%S")
287 
288             left_image_path = os.path.join(save_path, f"left_{timestamp}.png")
289             right_image_path = os.path.join(save_path, f"right_{timestamp}.png")
290 
291             wrist_left_image_path = os.path.join(save_path, f"wrist_left_{timestamp}.png")
292             wrist_right_image_path = os.path.join(save_path, f"wrist_right_{timestamp}.png")
293 
294             # 1.直接调用 write 方法，不一定有这个方法
295             left_image.write(left_image_path)
296             right_image.write(right_image_path)
297             wrist_left_image.write(wrist_left_image_path)
298             wrist_right_image.write(wrist_right_image_path)
299 
300             # 2.手动存
301             # 转换为 BGR 格式以便用 OpenCV 保存
302             # left_image_bgr = cv2.cvtColor(left_image, cv2.COLOR_RGB2BGR)
303             # right_image_bgr = cv2.cvtColor(right_image, cv2.COLOR_RGB2BGR)
304             # wrist_left_image_bgr = cv2.cvtColor(wrist_left_image, cv2.COLOR_RGB2BGR)
305             # wrist_right_image_bgr = cv2.cvtColor(wrist_right_image, cv2.COLOR_RGB2BGR)
306 
307             # cv2.imwrite(left_image_path, left_image_bgr)
308             # cv2.imwrite(right_image_path, right_image_bgr)
309             # cv2.imwrite(wrist_left_image_path, wrist_left_image_bgr)
310             # cv2.imwrite(wrist_right_image_path, wrist_right_image_bgr)
311 
312             # print(f"Images saved: {left_image_path}, {right_image_path}")
313 
314         elif state == "normal":
```

注：

```
295             # 1.直接调用 write 方法，不一定有这个方法
296             left_image.write(left_image_path)
297             right_image.write(right_image_path)
298             wrist_left_image.write(wrist_left_image_path)
299             wrist_right_image_path.write(wrist_right_image_path)
300
301             # 2.手动存
302             # 转换为 BGR 格式以便用 OpenCV 保存
303             # left_image_bgr = cv2.cvtColor(left_image, cv2.COLOR_RGB2BGR)
304             # right_image_bgr = cv2.cvtColor(right_image, cv2.COLOR_RGB2BGR)
305             # wrist_left_image_bgr = cv2.cvtColor(wrist_left_image, cv2.COLOR_RGB2BGR)
306             # wrist_right_image_bgr = cv2.cvtColor(wrist_right_image, cv2.COLOR_RGB2BGR)
307
308             # cv2.imwrite(left_image_path, left_image_bgr)
309             # cv2.imwrite(right_image_path, right_image_bgr)
310             # cv2.imwrite(wrist_left_image_path, wrist_left_image_bgr)
311             # cv2.imwrite(wrist_right_image_path, wrist_right_image_bgr)
```

如果这两句话报错，没有 write 这个方法的话。

```
1  left_image.write(left_image_path)
2  right_image.write(right_image_path)
```

就把这 4 行注释掉，把下面 8行的注释取消，手动保存，不调用这个 write 方法

`ZEDCamera.py` 文件

在 `gello_software/experiments` 文件夹下，添加 `ZEDCamera.py`

> 这个 ZED SDK 怎么连接相机我还没搞懂，这个`ZEDCamera.py` 实现了 ZedCamera类
>
> 这个类很简单只有三个函数：
>
> 1.`__init__`： 完成相机初始化
>
> 2.`capture_frame`：获取左右相机图像，返回 left_image， right_image
>
> 3.析构函数，释放相机实例

这个还得看看怎么连接相机，然后可能还需要再做一定的修改。

反正大框架是确定了，只要实现这三个函数，在`run_env.py` 中我已经手动写好了保存视频的代码，从268-313行的位置。

然后在下面的`convert_data_to_libero.py` 将数据转换到 `LeRobot` 格式，我也是按照这个 `run_env.py` 的实现风格，从主视角和腕部相机中加载数据进行转换。

`convert_data_to_libero.py`

预想的数据路径：

```
bc_data
-----gello
----------时间戳1
----------------image文件夹（存放主相机的帧）
----------------wrist_image文件夹（存放腕部相机的帧）
----------------example1.pkl 文件
----------------example2.pkl 文件
-----------------......
----------时间戳2
```

> 这个脚本实现了
>
> 从 `bc_data/gello` 下的所有文件，以时间戳为单位读取文件转换成 LeRobot 格式

```
from lerobot.src.lerobot.datasets.lerobot_dataset import LeRobotDataset
```

这个文件要添加在 `openpi` 文件夹下，然后 23 行，也就是上面这一行的导入路径可能还要根据放的位置再修改一下。

这个 `convert_data_to_libero.py` 文件还可能需要再修改，还没有 debug

我还打算写一个从 droid 格式下读取数据的脚本，既可以用 `openpi/examples/droid/main.py` 来训练，也可以走 Libero 那边，用 `openpi/examples/libero/main.py` 来训练。

明天再写吧，休息了