Training CrazyFile 2.1 micro drone using PPO

A custom implementation of a physical environment to train a Crazyflie drone using PPO. The script is designed for use with the Crazyflie 2.X micro drone and employs the `PPO` algorithm from the `Stable-Baselines3` library for training.

Features

1. **Custom Gym Environment**:
    - `CrazyflieEnv`: Base class for simulating and controlling the Crazyflie drone.
    - `PhysicalCrazyflieEnvWrapper`: Extends the base environment with additional safety features, wait signal for physical reset, auto reconnect on restart, reward mechanisms, and emergency stop functionality.
2. **Reinforcement Learning**:
    - Utilizes Proximal Policy Optimization (PPO) for training.
    - Supports saving and resuming models, logging, and using replay buffers.
3. **Safety Features**:
    - Emergency stop functionality (`Ctrl+C`).
    - Position, velocity, and stability-based reward system.
    - Limits on drone movement to ensure safe operation.
4. **Logging and Monitoring**:
    - Logs Crazyflie sensor data (position, roll, pitch, yaw) for real-time updates.
    - Tensorboard support for monitoring training progress.

Requirements

- **Python Libraries**:
    - `gym`
    - `stable-baselines3`
    - `numpy`
    - `keyboard`
    - `cflib` (Crazyflie Python API)
    - `matplotlib` (for optional visualization)
- **Hardware**:
    - Crazyflie 2.X drone.
    - Crazyradio PA for communication.
- **System Configuration**:
    - Ensure the drone is in a safe, open space before starting.
    - Install and set up the Crazyflie drivers using the `cflib` library.

Usage

Start a New Training Session

Run the script without arguments to start training a new PPO model:

```
python train_crazyflie.py
```

Resume Training from a Saved Model

To resume training from a previously saved model:

```
python train_crazyflie.py <path_to_model>
```

Emergency Stop

Press `Ctrl+C` during training to trigger an emergency stop and safely land the drone.

Environment Details

Observation Space

- **State Vector**: `[x, y, z, roll, pitch, yaw]`
- Shape: `(9,)`

Action Space

- **Action Vector**: `[thrust, roll, pitch, yaw]`
- Type:

```
spaces.Box(low=np.array([-1, -1, -1, -1]),
           high=np.array([1, 1, 1, 1]),
           dtype=np.float32            )
```

Rewards

- Proximity to the target position.
- Stability in roll, pitch, and yaw.
- Penalization for out-of-bounds or unsafe conditions.

Hyperparameter Configuration

Default parameters used for PPO training:

- `learning_rate`: 0.005
- `n_steps`: 256
- `batch_size`: 16
- `n_epochs`: 5
- `gamma`: 0.98
- `gae_lambda`: 0.95

Logging and Visualization

- Checkpoints are automatically saved during training.
- Tensorboard logs are saved in the `cache` directory. Run the following command to visualize:

```
tensorboard --logdir=cache
```