First Step

- 首先实现一个 flocking 并使用 PID 控制， 运动方式由路线点给出的 简单 demo.
- 所有的无人机都由 reynold 控制即可.

Second Step

- 此处需要实现关于 无人机偏航角的 实现.

drone states

```
state = np.hstack([
            self.pos[nth_drone, :], self.quat[nth_drone, :], # 3,4
            self.rpy[nth_drone, :], self.vel[nth_drone, :],  # 3,3
            self.ang_v[nth_drone, :],
self.last_clipped_action[nth_drone, :]
        ])
```

utils

- [x] 性能分析 https://blog.csdn.net/weixin_40583722/article/details/121659851

```
python -m cProfile -o flame_of_flocking.prof flocking.py
flameprof flame_of_flocking.prof > flame_of_flocking.svg
```

PROF

- [x] 使用tensorboard

```
tensorboard --logdir=/home/lih/fromgit/gym-pybullet-
drones/gym_pybullet_drones/src/results/
```

TODO LIST

- [x] 需要在 flocking_aviary 中将 ctrl_freq 和 decision_freq 分离开来，因为 BaseAviary 中 step() 是按照 ctrl_freq 调用的
- [x] 计算 gaussian_process 需要的计算压力太大了， 需要提高运算速度， 考虑使用 `GPyTorch`
  - 使用了 GPyTorch, 将运行速度提升了 5 倍
- [x] 考虑使用 方位测量 和 距离测量进行替代, 这样的 reward 似乎难以设计

- [x] 使 control_by_RL_mask 的无人机无法接受迁移控制指令
- [x] 考虑如何设计 reward function， 是否把对 angular speed 的幅度限制加入 `_computeTerminated()`，已加入
- [x] 为 gptorch 添加数据的归一化与反归一化，现在 gpytorch 的行为与 sklearn 差异过大 已解决

几点初步设计思路

- [x] 当前 control by RL 的无人机无法获取迁移指令？ 这合理吗，容易在初期丢失目标造成 terminated (这是在 eval 时观察的情况)
    - 并不是 eval 时没有获得迁移指令，而是 determinstic 模式下无法运动
    - eval时，没有将 fov
- [x] reward 完全无法得到收敛，动作空间的设计方式是否有问题!
- [] 考虑应当模仿图的离散化，重新设计action_space更小，
    - [x] 考虑将 action_space 设计为基于 diff 的形式 , 这一解法之前有误，现已解决。 目前控制赶不上规划
    - [x] 使用 speed 模式时，无法产生真实 yaw action
    - [ ] 考虑重新设计 reward, 我这个持续监控的模式，可能不适合reward
- [x] 将 num_uav 和 control_by_RL_mask 设置为随机
- [x] 减少 action space, 提前 truncted RL

STAMP 的新思路

- [x] 探讨了各种建模对于环境的影响，但无论如何，我们需要对 action space 进行图的离散化。
    - [x] 这里需要考虑的是如何评估 Node_feature，这里需要将 GP 建模在 action space 之中
    - [x] 先调通在图的离散化中基本的操作
- [x]使用 IPP 思路进行建模， action type 应当为 yaw 直接控制的形式

先考虑简化OBS与网络的设计，

- [] budget 在 IPP 中用于判断下一个节点是否可达，我似乎并不需要这样一个 budget
    - [x] 这个 budget 可以用于现在在 knn 中的连接，因为我们需要限制过大的 yaw action

PROF

- [x] 那么现在输入为增广图， 已执行过的路径，这是一个简化
- [x] 按照我的理解 ，我的 action_net 与 value_net 已经包含在 attention_net 之中
- [x] 和每一个 node 连接的节点数量需要多一点，以能够再次发现目标
- [x] 先减少训练时，无人机的数量，
- [ ] 还是需要考虑 FOV 对不确定性的影响, 这里也是
- [ ] 如何建模 reward function？ 归一化？

Install

- 需要自定义 pythonpath 避免 gym 使用已经注册并移动至 sitepackages 目录的环境：

```
# set python path
export PYTHONPATH=/home/lih/fromgit/gym-pybullet-drones:$PYTHONPATH
echo set python path for flocking aviary done!
```

---

eval/	
mean_ep_length	439
mean_reward	192
time/	
total_timesteps	32000
train/	
approx_kl	0.012960765
clip_fraction	0.0593
clip_range	0.2
entropy_loss	-2.27
explained_variance	1.19e-07
learning_rate	0.0003
loss	24.3
n_updates	30
policy_gradient_loss	-0.00554
value_loss	72.2

New best mean reward!
Eval num_timesteps=64000, episode_reward=93.09 +/- 96.29
Episode length: 341.20 +/- 125.38

eval/	
mean_ep_length	341
mean_reward	93.1
time/	
total_timesteps	64000
train/	
approx_kl	0.008667494
clip_fraction	0.0284
clip_range	0.2
entropy_loss	-2.23
explained_variance	0
learning_rate	0.0003
loss	22.3
n_updates	70
policy_gradient_loss	-0.00169
value_loss	50.4

PROF

Eval num_timesteps=96000, episode_reward=141.64 +/- 242.13
Episode length: 299.60 +/- 327.63

eval/	
mean_ep_length	300
mean_reward	142
time/	
total_timesteps	96000
train/	
approx_kl	0.01023529
clip_fraction	0.0369
clip_range	0.2
entropy_loss	-2.25
explained_variance	-1.19e-07
learning_rate	0.0003
loss	30.1
n_updates	110
policy_gradient_loss	-0.00225
value_loss	80.1

Eval num_timesteps=128000, episode_reward=125.17 +/- 93.86
Episode length: 434.80 +/- 197.93

eval/	
mean_ep_length	435
mean_reward	125
time/	
total_timesteps	128000
train/	
approx_kl	0.009777371
clip_fraction	0.0435
clip_range	0.2
entropy_loss	-2.24
explained_variance	0
learning_rate	0.0003
loss	39.3
n_updates	150
policy_gradient_loss	-0.00237
value_loss	85.2

Eval num_timesteps=160000, episode_reward=170.10 +/- 273.77
Episode length: 486.00 +/- 270.82

eval/	
mean_ep_length	486
mean_reward	170

time/	
total_timesteps	160000
train/	
approx_kl	0.008861544
clip_fraction	0.0219
clip_range	0.2
entropy_loss	-2.18
explained_variance	-1.19e-07
learning_rate	0.0003
loss	16.5
n_updates	190
policy_gradient_loss	-0.00133
value_loss	44.6

Eval num_timesteps=192000, episode_reward=97.40 +/- 111.73
Episode length: 262.20 +/- 142.86

eval/	
mean_ep_length	262
mean_reward	97.4
time/	
total_timesteps	192000
train/	
approx_kl	0.00839573
clip_fraction	0.029
clip_range	0.2
entropy_loss	-2.19
explained_variance	1.19e-07
learning_rate	0.0003
loss	33.3
n_updates	230
PROF	policy_gradient_loss	-0.0019
value_loss	66.3

Eval num_timesteps=224000, episode_reward=143.31 +/- 153.38
Episode length: 411.60 +/- 128.02

eval/	
mean_ep_length	412
mean_reward	143
time/	
total_timesteps	224000
train/	
approx_kl	0.016349137
clip_fraction	0.0949
clip_range	0.2

entropy_loss	-2.04
explained_variance	0
learning_rate	0.0003
loss	30.4
n_updates	270
policy_gradient_loss	-0.00658
value_loss	53

Eval num_timesteps=256000, episode_reward=309.00 +/- 425.16
Episode length: 333.60 +/- 308.64

eval/	
mean_ep_length	334
mean_reward	309
time/	
total_timesteps	256000
train/	
approx_kl	0.0104342215
clip_fraction	0.0249
clip_range	0.2
entropy_loss	-2.07
explained_variance	0
learning_rate	0.0003
loss	11.9
n_updates	310
policy_gradient_loss	-0.00158
value_loss	43.1

New best mean reward!
Eval num_timesteps=288000, episode_reward=45.31 +/- 47.75
Episode length: 279.00 +/- 87.62

PROF

eval/	
mean_ep_length	279
mean_reward	45.3
time/	
total_timesteps	288000
train/	
approx_kl	0.014717518
clip_fraction	0.064
clip_range	0.2
entropy_loss	-2
explained_variance	5.96e-08
learning_rate	0.0003
loss	15.1
n_updates	350

| policy_gradient_loss | -0.00492 |
| value_loss | 57.1 |