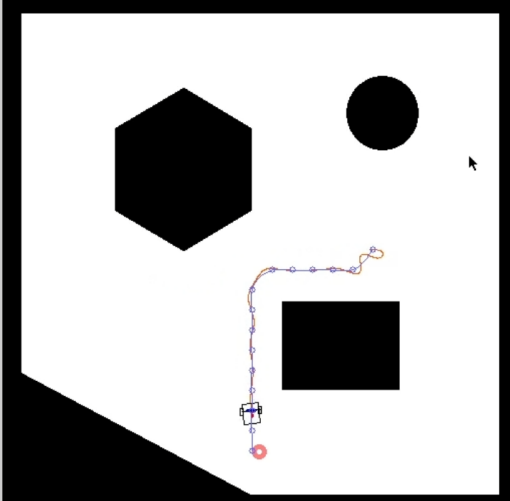
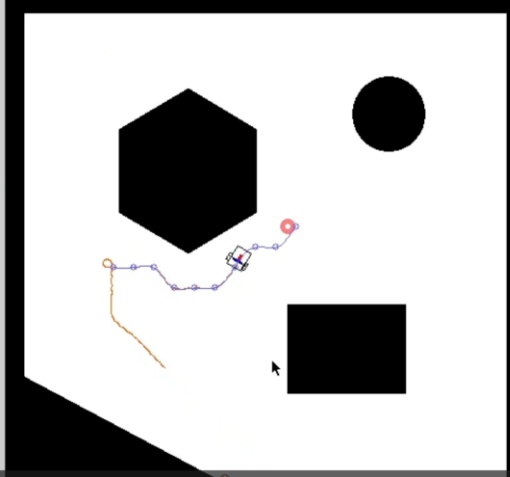
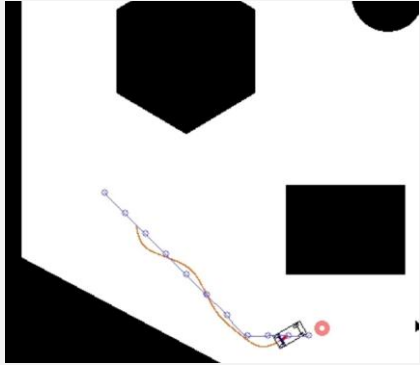
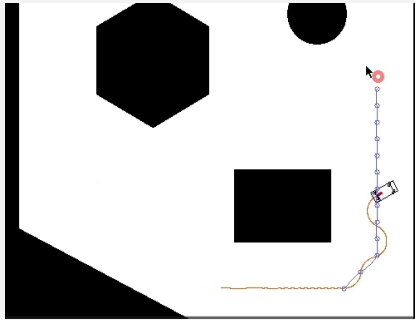
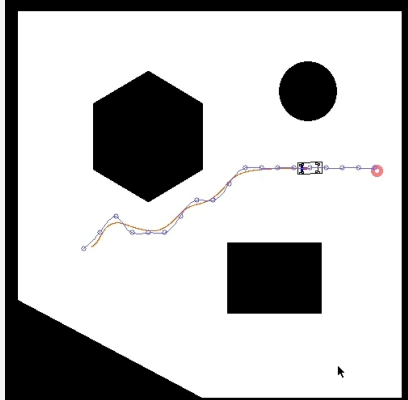


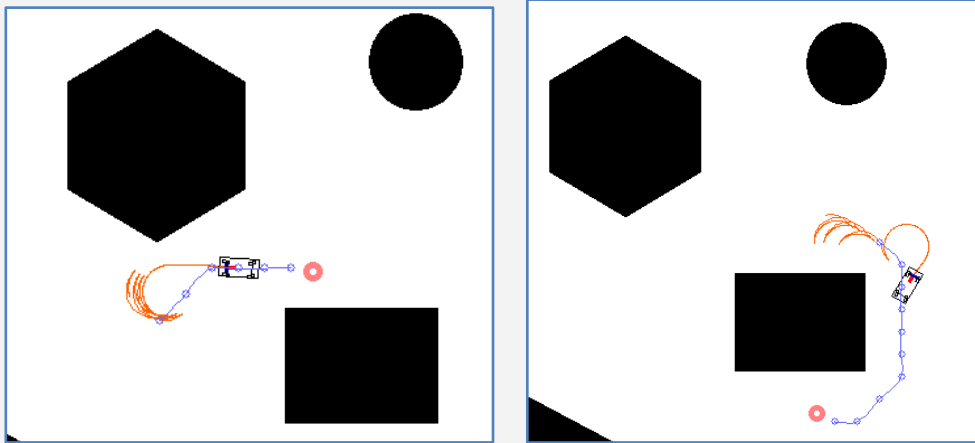
1. kinematic models: basic / differential drive

pid		<p>PID算法中最重要的是把P gain, I gain 和 D gain 調到合理值，最科學的作法是用Matlab調PID。</p> <p>但現在就看圖手調參數。</p> <p>調整後，某些pathtracking的震幅還是會很大，但大部分是可以接受的範圍。</p> <p>另外只對角速度做了PID，直線速度只給定值。</p>
pure pursuit		<p>Pure Pursuit 很明顯地改善PID偶爾會有比較大steady state error問題，但伴隨而來的問題是 tracking oscillation，即小幅震盪和急遽抖動/轉向。</p> <p>雖然pathtracking的目標本身達成，但現實層面會讓機器控制難以穩定、乘客體感十分不舒服。</p>
lqr		

2. kinematic models: bicycle

pid		<p>bicycle model 放大了在 differential drive demo中的問題，總是會有個不容忽視的穩態誤差。</p> <p>但也可能是PID參數調得太爛了。</p> <p>另外因為 bicycle model 是控制 a 和 δ，δ 由 PID 給出，a 則是自己設的速度上限減去當前的車速。</p>
pure pursuit		<p>在左圖中，可以看到第一個目標的 path tracking 的穩態誤差很小；但是第二個目標點卻甚至比PID還大。在少數情況下還會原地打轉。</p> <p>我有嘗試不同的 lookahead distance，但似乎沒有太多的改變。</p> <p>想了很久但還是個未解之謎。</p>
stanley		<p>Stanley 得出的 tracking path 就漂亮且穩定很多。但也花了不少時間算數學...</p>
lqr		

3. collision



邏輯：撞牆後先倒退，重新嘗試 pathtracking，如此反覆超過一定次數後就做 path replanning，re-plan的方法很樸素，用第一個作業的cubic spline對當前路徑做圓滑。

感想：大部分情況下都不會到 re-plan 的環節，通常倒退幾次後就能闖出去。
用 bicycle model x stanley control 比較好觀察到 collision handling 的邏輯。

備註：運行環境為 docker

```
Dockerfile x  compose.yml M  planner_a_star.py 2, U  path_planning.py 2, U  planner.py 2, U
1 FROM python:3.10-slim
2
3 ENV DEBIAN_FRONTEND=noninteractive
4
5 RUN apt update && apt install -y \
6     libgl1 \
7     libglu1-mesa-dev \
8     libgtk2.0-dev \
9     && rm -rf /var/lib/apt/lists/*
10
11 RUN pip install \
12     opencv-python \
13     numpy
14
15 CMD [ "/bin/bash" ]

compose.yml
1 docker-compose.yml - The Compose specification establishes a standard for the definition of multi-container
2 #####
3 ### shared settings ###
4 #####
5 x-common-settings: &common
6
7 build:
8   context: .
9   dockerfile: Dockerfile
10  image: pomelo925/2025-spring-course:robotic-nav-exploration
11
12 volumes:
13   # GUI
14   - $HOME/.Xauthority:/root/.Xauthority
15   - /tmp/.X11-unix:/tmp/.X11-unix
16
17   # workspace
18   - ../110033226_HW1:/110033226_HW1
19
20 environment:
21   - DISPLAY=${DISPLAY}
22
23 tty: true
24 network_mode: host
25 privileged: true
26 stop_grace_period: 1s
27
28 #####
29 ### Container Services ###
30 #####
31 > Run All Services
32
33 services:
34   > Run Service
35   robot:
36     build:
37       <<: [*common]
38     container_name: robotic-nav-exploration
39     command: ["/bin/bash"]
```