Formulation of the problem

Chang Liu

I. Introduction

My work can different from Charrow's work in the following aspects: 1. Instead of fitting each particle with a GMM, I generate a GMM with much smaller component number for planning purpose. 2. In the planning, I incorporate the effects of limited FOV. 3. In the path planning, I use MPC that may incorporate uncertainties.

I should mention that, strategies that only updates GMM weights are not suitable, since the GMM here is not for mode that is fixed, but for dynamically changing states. Therefore, it is important to update the mean and covariance.

II. PROBLEM FORMULATION

A. Robot and Target Motion Model

Unicycle motion model for the mobile robot:

$$z_{k+1} = h(z_k, u_k^r), \tag{1}$$

where

$$f(z_k, u_k^r) = z_k + \begin{bmatrix} \cos\theta_k^r \Delta t & 0 \\ \sin\theta_k^r \Delta t & 0 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} u_k^r.$$

Motion model of the target:

$$x_{k+1} = f(x_k) + w_k,\ w_t \sim \mathcal{N}(0, Q) \tag{2}$$

$$\tag{3}$$

B. Modeling Sensing Domain

Sensor sensing domain is represented as $\mathcal{F}_k = \big\{[x_{1,k}, x_{2,k}] \in \mathbb{R}^2 |\ \|v\|_2 \le r, \angle v \in [\theta_1, \theta_2]\big\}$, where $v = [x_{1,k} - z_{1,k}, x_{2,k} - z_{2,k}]$.

C. Sensor Measurement Model

Measurement model:

$$y_k = g(x_k) + v_k,\ v_k \tag{4}$$

It has four modes,

if $\gamma_k = 1$

$$P(Y_k|x_k) = \begin{cases} P(y_k|x_k) \sim \mathcal{N}(g(x_k), R) \\ P(\emptyset|x_k) = 0 \end{cases} \tag{5}$$

if $\gamma_k = 0$

$$P(Y_k|x_k) = \begin{cases} P(y_k|x_k) = 0 \\ P(\emptyset|x_k) = 1 \end{cases} \tag{6}$$

where $\gamma_k = \mathbb{1}_{\{x_k \in \mathcal{F}_k\}}$.

III. MPC-based Path Planning

A. EKF with Limited Sensing Domain

$$\hat{x}_{k+1|k}^t = f(\hat{x}_{k|k}^t) \tag{7a}$$
$$P_{k+1|k} = A_k^i P_{k|k}^i A_k^{i'} + Q \tag{7b}$$
$$K_{k+1}^i = P_{k+1|k}^i C_{k+1}^{i'} (C_{k+1}^i P_{k+1|k}^{i'} C_{k+1}^{i'} + R)^{-1} \tag{7c}$$
$$\hat{x}_{k+1|k+1}^i = \hat{x}_{k+1|k}^i + \gamma_{k+1} K_{k+1}^i (y_{k+1} - h(\hat{x}_{k+1|k}^t)) \tag{7d}$$
$$P_{k+1|k+1}^i = P_{k+1|k}^i - \gamma_{k+1} K_{k+1}^i C_{k+1}^i P_{k+1|k}^i, \tag{7e}$$

where $A_k^i = \frac{\partial f}{\partial x}|_{x=\hat{x}_{k|k}^i}$ and $C_{k+1}^i = \frac{\partial g}{\partial x}|_{x=\hat{x}_{k+1|k}^i}$. The $\hat{x}_{k|k}^t$ and $P_{k|k}$ represent the estimated target position and covariance matrix. For notational simplicity, we define $b_k = [\hat{x}_k^t, P_{k|k}]$ and let $b_{k+1} = g(b_k, u_k^r)$ represent the Kalman filter defined in Eq. (7).

γ is approximated by

$$\gamma_k \approx \frac{1}{1 + \alpha_1 \|[x_{1,k}, x_{2,k}] - [z_{1,k}, z_{2,k}]\|_2^2} \times \frac{1}{1 + \exp\Big\{-\alpha_2(\cos(\theta_k^r - \tilde{\theta}_k) - \cos(\theta_0))\Big\}}, \tag{8}$$

where $\tilde{\theta}_k = \angle([x_{1,k}, x_{2,k}] - [z_{1,k}, z_{2,k}])$ is the direction angle from the sensor position to target position; $\theta_0 = \frac{\theta_2 - \theta_1}{2}$ is half of the sensing angle; α_1 and α_2 are tunning parameters that controls the shape of the function. Eq. (8) can be interpreted as follows: when the robot is close to the target, it is more likely that the target can be detected; besides, the closer the target direction aligns with the center direction of the sensor, the higher possibility that the target will get detected.

B. Path Planning for Target Search and Tracking

The MPC-based path planner with planning horizon N can be formulated as:

$$\min_{u_{1:N}} J(b_{1:N+1}, u_{1:N}) \tag{9a}$$
$$\text{s.t.}\ z_{k+1} = f(z_k, u_k^r), \tag{9b}$$
$$b_{k+1} = g(b_k, u_k^r), \tag{9c}$$
$$z_{k+1} \in \mathcal{X},\ u_{k+1}^r \in \mathcal{U}, \tag{9d}$$
$$k = 1, \dots, N, \tag{9e}$$

The objective function is

$$J(b_{1:N+1}) = \sum_{k=1}^{N+1} H(b_k) \tag{10}$$

$$\approx -\sum_{k=1}^{N+1}\sum_{i=1}^{L} w_i \log \sum_{j=1}^{L} \mathcal{N}(\hat{x}^i_{k|k}|\hat{x}^j_{k|k}, P^j_{k|k}). \tag{11}$$

The approximation is the 0-order approximation of entropy.

C. Possible linearization in the iterative planning process

1) target motion matrix A
2) sensor measurement matrix C
3) initial solution
4) approximate the sensor boundary
5) linearize robot motion model

D. estimation-planning framework

In the estimation based on the actual measurement, use the particle filter to incorporate nonlinearity and use E-M at each step to fit a GMM. mention that we can use other advanced technologies to update GMM, but here we just use a simple one.

In the planning, use γ to update the estimation. This makes sense since if we use pseudo-measurement to update GMM, it only reinforces/bootstrapping the current GMM, brining no new info. So it makes sense to just consider the entropy of the uncertainty.

In the MPC, I may need to include the uncertainty into the obj. It seems that the only uncertainty comes from γ in Eq. (7d) and (7e). The idea is that, since γ is dependent on the underlying distribution of $x^i_{k+1|k}$, we can either assume MAP for the target position or use integral to compute the expected update of Eq. (7d) and (7e).

IV. Sequential optimization framework

The problem uses two-layer of optimization. In the first layer, a non-convex problem is solved and the solution is provided to the second layer, which uses a convex optimization. (NOTE: may need to compare with iterative LQR).

A. nonconvex programming

The purpose of this layer is to find a good reference state for cvxPlanner to linearize around. The formulation uses the one defined in Eq. (9).

simplification:

- objective: the covariance matrix is assumed constant in planning horizon.
- weights of gmm are assumed constant.
- mean propagation: open-loop using the system dynamics, no measurement is used.
- covariance propagation: compute the gamma assuming the mean of a gmm component as the target position to update the covariance of the corresponding gmm component. $\bar{\theta}$ takes the value of the initial guess.

Initial solution:

- At the beginning of the planning, use straight line constant acceleration model to generate initial guess. Later can change to motion primitives and try multiple initial guesses (similar to what John Schulman's work does).
- When cvxPlanner is used, use the solution of cvxPlanner with one-step extrapolation as the initial guess for ngPlanner in next round.

Possible changes:

- whether using initial guess to approximate A, C.
- it is unclear to me whether I should us ngPlanner at all (after the first step). I may be able to use cvxPlanner only for the rest of whole search process.

B. iterative convex programming formulation

This layer uses sequential convex programming to plan a trajectory.

Simplification:

- use the convex relaxation of 0-th order approximation of entropy in the objective function.
- linearize eq. (9b) around reference trajectory.
- linearize eq. (9c) around reference trajectory. Especially, the K is assumed to use the reference value.
- A,C all uses the value of reference state.

possible changes:

- the sensor FOV approximation is bad. only gives value 1 in very small state range.

1) Approximating the Objective Function: We can obtain the upper bound of Eq. (11) using Jensen's inequality:

$$\begin{aligned}
&-\sum_{k=1}^{N+1}\sum_{i=1}^{L} w_i \log \sum_{j=1}^{L} \mathcal{N}(\hat{x}^i_{k|k}|\hat{x}^j_{k|k}, P^j_{k|k}) \\
&\leq -\sum_{k=1}^{N+1}\sum_{i=1}^{L} w_i \sum_{j=1}^{L} \log \mathcal{N}(\hat{x}^i_{k|k}|\hat{x}^j_{k|k}, P^j_{k|k}) \\
&= \sum_{k=1}^{N+1}\sum_{i=1}^{L} w_i \sum_{j=1}^{L} (\hat{x}^i_{k|k} - \hat{x}^j_{k|k})^T [P^j_{k|k}]^{-1} (\hat{x}^i_{k|k} - \hat{x}^j_{k|k})
\end{aligned}$$

The upper bound is tight when all GMM components become equivalent.

Note that this is NOT a quadratic objective function, but it is convex! (matrix-fractional function). This can be transformed into LMI as this:

$$(\hat{x}^i_{k|k} - \hat{x}^j_{k|k})^T [P^j_{k|k}]^{-1} (\hat{x}^i_{k|k} - \hat{x}^j_{k|k}) \leq t \iff$$
$$\begin{bmatrix} P^j_{k|k} & \hat{x}^i_{k|k} - \hat{x}^j_{k|k} \\ (\hat{x}^i_{k|k} - \hat{x}^j_{k|k})^T & t \end{bmatrix} \geq 0, \qquad t \geq 0$$

V. Possible extensions

This section compiles the possible contributions I can make in this and other following works. The previous sections are the actual implementation of what I will do in this work.

1) GSF with good weight and component number update law

    a) GMM-PF: update PF and GMM at the same time, not running E-M to fit GMM for PF. One benefit of this is that we can always check the difference between PF and GMM and generate a better GMM if difference is big.
    b) when updating GMM, can we treat $\gamma = 0$ as a rare event? Will Dirichlet process help?
    c) treat $\gamma = 0$ and $\gamma = 1$ as hybrid system. Then use GMM to model the mode switch? I think a little bit about this but seems difficult to do.

2) efficient computation of the objective function
3) how to represent and compute γ (incorporating b_t or just using a point estimate (e.g. MAP))
4) incorporate negative info in a better way than γ.
5) the way to do iterative planing: updating γ in SQP or outside SQP; whether updating w. if not updating, can I obtain an upper bound of the error?
6) control of nonholonomic vehicle
7) make the problem a cvx optimization or some other form (e.g., proximal gradient) to better utilize the form of the problem.

1,3,4 are the possible main contributions of the work.