A predictive method for full-pose predictive distributed leader-follower formation control for non-holonomic robots

Titouan Renard, 2021

May 14, 2021

1 Problem Statement

We have a team of N differential-wheeled robots $\mathcal{R}_1, ..., \mathcal{R}_n$ described by the kinematic equations (the robot's dynamics are neglected) :

$$\dot{f}(\vec{x}, \vec{u}) = \begin{cases} \dot{x}_i = u_i \cos\theta_i \\ \dot{y}_i = u_i \sin\theta_i \\ \dot{\theta}_i = \omega_i \end{cases} . \tag{1}$$

Where $\vec{u}_i = [u_i, \omega_i]^T$ is the control input vector of $\mathcal{R}_i$ with u_i linear translational speed and ω_i rotational speed. And where $\vec{x}_i = [x_i, y_i, \theta_i]^T$ is the pose vector of $\mathcal{R}_i$. We denote the full pose and control input of the system as :

$$\vec{x} = [x_1, y_1, \theta_1, x_2, y_2, \theta_2, ..., x_N, y_N, \theta_N]^T \tag{2}$$

$$\vec{u} = [u_1, \omega_1, u_2, \omega_2, ..., u_N, \omega_N]^T \tag{3}$$

We denote $\mathbf{R} = \{\mathcal{R}_1, ..., \mathcal{R}_n\}$ the set of all robots. Each robot $\mathcal{R}_i$ has a set of *neighboring robots* $\mathcal{N}_i \subseteq \mathbf{R}$, which contains the set of robots for which $\mathcal{R}_i$ can get a position estimation. The pose of $\mathcal{R}_j$ estimated by $\mathcal{R}_i$ is given by a range ρ_{ij} and a bearing α_{ij}. Each pose estimation is affected by noise which is denoted ϵ_z and is denoted by a vector :

$$z_{ij} = \begin{bmatrix} \tilde{\rho}_{ij} \\ \tilde{\alpha}_{ij} \end{bmatrix} = \begin{bmatrix} \rho_{ij} \\ \alpha_{ij} \end{bmatrix} + \epsilon_z. \tag{4}$$

At time t robot $\mathcal{R}_i$ gathers an observation list :

$$\mathcal{Z}_i = \{z_{ij} |\ \mathcal{R}_| \in \mathcal{N}_i\}. \tag{5}$$

Our goal is to have robots $\mathcal{R}_2, ..., \mathcal{R}_N$ (that we call *followers*) maintain formation with the robot $\mathcal{R}_1$ (which we call *leader*) while avoiding obstacles in their trajectories. We look for a control law that can be implemented in a distributed fashion for robots $\mathcal{R}_2, ..., \mathcal{R}_N$, while the control law of $\mathcal{R}_1$ is defined arbitrarily.

The formation is defined by a set of *biases* $\vec{\beta}_i = [\delta_i^x, \delta_j^x, \delta_i^\theta]^T$, $\forall i = 2...N$ which denotes the expected pose of $\mathcal{R}_i$ relative to $\mathcal{R}_1$ within the formation. We can thus express the *pose error* $\bar{x}_i$ for $\mathcal{R}_i$ as :

$$\bar{x}_i = \vec{x}_i - \vec{\beta}_i = \begin{bmatrix} x_i - \delta_i^x \\ y_i - \delta_i^y \\ \theta_i - \delta_i^\theta \end{bmatrix} \tag{6}$$

The problem of maintaining formation thus becomes the problem of reducing the *total pose error* $\mathcal{E} = \sum_{2...N} \|\vec{e_i}\|$ in a distributed fashion.

2 Laplacian-based feedback for formation control

Let $G = (\mathbf{R}, \mathbf{E})$ be an undirected graph constructed such that

1. it's vertex set $\mathbf{R} = \{\mathcal{R}_1, ..., \mathcal{R}_n\}$ contains every single robot in the team
2. it's edges set contains an arbitrarily oriented edge for each robot in line of sight of another

$$\mathbf{E} = \{(\mathcal{R}_i, \mathcal{R}_j) | \ \mathcal{R}_j \in \mathcal{N}_i\}.$$

Let $\mathcal{I}$ denote the *incidence* matrix (with arbitrary orientations) of G and $\mathcal{W}$ it's weight matrix. We compute the *weighted laplacian matrix* of G as follows :

$$\mathcal{L} = \mathcal{I} \cdot \mathcal{W} \ \cdot \mathcal{I}^T$$

Note that the *weighted laplacian matrix* $\mathcal{L}$ is constructed in such a way that :

$$\dot{\vec{x}} = -\mathcal{L}x(t) \tag{7}$$

$$\dot{x_i} = \sum_{\mathcal{R}_j \in \mathcal{N}_i} w_{ij}(x_j - x_i) \tag{8}$$

A standard approach to formation control is to implement a Laplacian based feedback equation (which can be tough of as a PI controller) such as:

$$\dot{x} = -\mathcal{L}\bar{x} + K_I \int_0^t \mathcal{L}(\tau)\bar{x}(\tau)d\tau \tag{9}$$

2.1 Predictive approach to the laplacian-based formation control problem

We propose to apply an optimization based predictive control law to our robots :

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & J(\vec{u}) = \int_t^{t+\tau} L(\tau, \vec{x}, \vec{u}) + V(t+T, \vec{x}) \\ \text{subject to} \quad & \dot{f}(\vec{x}, \vec{u}) \end{aligned} \tag{10}$$

3 Theoretical analysis

In this section, we will use a *normalized* Laplacian matrix $\bar{\mathcal{L}} \in \mathbb{R}^{n\times n}$ (also called the random walk normalized Laplacian). Suppose we are given the standard Laplacian matrix $\mathcal{L}$ whose diagonal elements are represented by $D = \text{diag}(\mathcal{L}) \in \mathbb{R}^{n\times n}$. Then, the normalized Laplacian matrix is defined as

$$\bar{\mathcal{L}} = D^{-1}\mathcal{L}\,.$$

If the original Laplacian $\bar{\mathcal{L}}$ was unweighted, then the normalized Laplacian has the following form

$$\bar{\mathcal{L}}(i,j) = \begin{cases} 1 \text{ if } i = j\,, \\ -{}^1\!/_{d_i} \text{ for } i \neq j\,. \end{cases}$$

Here, d_i is the degree (number of neighbors) of the ith node. We summarize below some well known properties of the $\bar{\mathcal{L}}$ operator.

Claim 3.1 ([1, 2]). *The normalized Laplacian matrix satisifes*

1. *It is row stochastic satisfying* $\sum_{j=1}^n \bar{\mathcal{L}}(i,j) = 1$ *for all* $i \in [n]$.
2. *It has all real eigenvalues with the smallest being 0, of the form* $(\lambda_1 = 0, \lambda_2, \ldots, \lambda_n)$.
3. *The right eigenvector corresponding to eigenvalue 0 is the all 1s vector* $\frac{1}{\sqrt{n}}\mathbf{1}$.
4. *All the eigenvalues lie in range* $\lambda_1 \in [0,1]$ *and* $\lambda_2 > 0$ *iff the graph is connected.*

3.1 Laplacian based proportional response

Let us consider the simple proportional response algorithm with bias which performs the update

$$\dot{\boldsymbol{x}} = -\bar{\mathcal{L}}\boldsymbol{x} + \boldsymbol{b}\,. \tag{11}$$

Here, $\boldsymbol{b}$ encodes some bias about the final formation we want our system to converge to.

Claim 3.2. *Let $\boldsymbol{\pi}$ be the left eigenvector of the matrix $\bar{\mathcal{L}}$ corresponding to eigenvalue 0, and let $\boldsymbol{\pi}^\top \boldsymbol{b} = 0$. Then, assuming that $\lambda_2 > 0$ (i.e. the graph is connected) the update* (11) *coverges to the solution*

$$\boldsymbol{x}(\infty) = \alpha\mathbf{1} + \bar{\mathcal{L}}^\dagger \boldsymbol{b} \text{ where } \alpha = \frac{\boldsymbol{\pi}^\top \boldsymbol{x}(0)}{\boldsymbol{\pi}^\top \mathbf{1}}\,.$$

Further, the formation converges to this limit at an exponential rate

$$\|\boldsymbol{x}(t) - \boldsymbol{x}(\infty)\|_2 \leq \exp(-\lambda_2 t)\|\boldsymbol{x}(0) - \boldsymbol{x}(\infty)\|_2\,.$$

Proof. First note that all the eigenvalues of $-\bar{\mathcal{L}}$ lie in $[-1, 0]$, and so the trajectory of $\boldsymbol{x}(t)$ is stable. Now, since all the eigenvalues of $\bar{\mathcal{L}}$ are real, we can perform an SVD decomposition as follows

$$\bar{\mathcal{L}} = U\Lambda V^\top = 0(\mathbf{1}\boldsymbol{\pi}^\top) + \sum_{i=2}^{n} \lambda_i \boldsymbol{u}_i \boldsymbol{v}_i^\top\,,$$

where Λ is a diagonal matrix consisting of eigenvalues, and U and V are orthonormal matrices. Notice that since $\boldsymbol{\pi}^\top \bar{\mathcal{L}} = 0$, the $\boldsymbol{\pi}$ weighted average of $\boldsymbol{x}(t)$ remains constant over time:

$$\frac{d\boldsymbol{\pi}^\top \boldsymbol{x}}{dt} = \boldsymbol{\pi}^\top \dot{\boldsymbol{x}} = -(\boldsymbol{\pi}^\top \bar{\mathcal{L}})\boldsymbol{x} + \boldsymbol{\pi}^\top \boldsymbol{b} = 0\,.$$

This means that

$$\boldsymbol{\pi}^\top \boldsymbol{x}(t) = \boldsymbol{\pi}^\top \boldsymbol{x}(0)\,. \tag{12}$$

Next, observe that the only fixed point of the update equation (11) is of the form

$$\bar{\mathcal{L}}\boldsymbol{x}(\infty) = \boldsymbol{b} \Rightarrow \boldsymbol{x}(\infty) = \bar{\mathcal{L}}^\dagger \boldsymbol{b} + \alpha\mathbf{1}\,,$$

where $\bar{\mathcal{L}}^\dagger$ represents the pseudo-inverse and α is some constant. Using the previous constraint on the weighted sum of $\boldsymbol{x}(t)$, we can compute the value of α as

$$\boldsymbol{\pi}^\top \bar{\mathcal{L}}^\dagger \boldsymbol{b} + \alpha\boldsymbol{\pi}^\top \mathbf{1} = \alpha\boldsymbol{\pi}^\top \mathbf{1} \Rightarrow \alpha = \frac{\boldsymbol{\pi}^\top \boldsymbol{x}(0)}{\boldsymbol{\pi}^\top \mathbf{1}}\,.$$

This proves the first part of the claim. We can now solve the matrix differential equation above to write

$$\boldsymbol{x}(t) - \boldsymbol{x}(\infty) = \exp(-\bar{\mathcal{L}}t)(\boldsymbol{x}(0) - \boldsymbol{x}(\infty)) = (\exp(-\bar{\mathcal{L}}t) - \mathbf{1}\pi^\top)(\boldsymbol{x}(0) - \boldsymbol{x}(\infty))\,.$$

The last equality follows from our observation (12). Now, taking euclidean norms and both sides gives

$$\|\boldsymbol{x}(t) - \boldsymbol{x}(\infty)\|_2 \leq \left\|\exp(-\bar{\mathcal{L}}t) - \mathbf{1}\pi^\top\right\|_2 \|\boldsymbol{x}(0) - \boldsymbol{x}(\infty)\|_2 = \exp(-\lambda_2 t)\|\boldsymbol{x}(0) - \boldsymbol{x}(\infty)\|_2\,.$$

This finishes our proof.

□

References

[1] Radu Horaud. "A short tutorial on graph Laplacians, Laplacian embedding, and spectral clustering". In: *URl: http://csustan. csustan. edu/˜ tom/Lecture-Notes/Clustering/GraphLaplacian-tutorial. pdf* (2009).

[2] EL Wilmer, David A Levin, and Yuval Peres. "Markov chains and mixing times". In: *American Mathematical Soc., Providence* (2009).