Robots that Work Together
Robots that Play Together

Mac Schwager
Aeronautics and Astronautics
Stanford University
Distributed Optimization for Robots Working Together

Game Theoretic Planning for Robots “Playing” Together
Collaborate with Distributed Optimization
Compete with Game Theory
Distributed Optimization for Collaborating Robots

Abstract

In this paper, we consider the problem of distributed target tracking in a fleet of vehicles collaborating over a dynamic communication network, posed as a Maximum A Posteriori (MAP) optimization problem. Our key contribution is a scalable Distributed Rolling Window Tracking (DRWT) algorithm derived from the Alternating Direction Method of Multipliers (ADMM) distributed optimization framework. The algorithm consists of closed-form algebraic iterations approximating the centralized Kalman-like update such that each agent's estimate is independent of the communication network, and the agents can start from arbitrary initial conditions. We demonstrate our algorithm in a realistic urban driving scenario in the CARLA urban driving simulator. The proposed algorithm outperforms the Consensus Kalman Filter, and demonstrates large-scale simulations in a CARLA urban driving scenario.

I. INTRODUCTION

Several approaches have previously been applied to solving fundamental challenges on collaborative tracking. Given a network track the positions and velocities of 50 target vehicles autonomous cars connected on a time-varying communication network. Each sensing entity. The second author was funded on an NSF GRF, and the third on an NDSEG Fellowship.

Constraints on communication and computation impose limitations on the amount of information that can be disseminated in a distributed manner. A key challenge in integrating autonomous vehicles into the transportation infrastructure is ensuring their safe operation. The paper is organized as follows. We give related work in Sec. II and pose the distributed estimation problem in Sec. III. In Sec. IV, we formulate the centralized MAP estimation in the presence of potential hazards, such as human-occlusions. Collaborative estimation among networked autonomous vehicles has the potential to alleviate the limitations of autonomous cars and pedestrians. However, tracking the paths of these safety-critical targets using on-board sensors is difficult in urban environments due to the presence of occlusions. Collaborative estimation among networked autonomous vehicles can collectively improve the safety of their operating vehicles and pedestrians.
Motivation: Distributed Target Tracking
Distributed Optimization on a Graph

Separable Objective

\[
\min_{x \in \mathbb{R}} \sum_{i=1}^{n} J_i(x_i)
\]

- Optimize with iterations on each node
- Nodes communicate between neighbors in graph
Related Work

**Consensus Gradient Descent**

Tsitsiklis, J. *et al.* Distributed asynchronous deterministic and stochastic gradient optimization algorithms (1986)


**Consensus ADMM**

Mateos, G. *et al.* Distributed sparse linear regression (2010)

Boyd, S. *et al.* Distributed optimization and statistical learning via the alternating direction method of multipliers (2011)

**Other Augmented Lagrangian Methods**

Chen and Teboulle (1994)

Zhang and Zavlanos (2018)
Multiple Robots Tracking Multiple Dynamic Targets

Motion model with process noise:
\[ x^{t+1} = f(x^t, u^t) + w^t \]
\[ w^t \sim N(0, Q) \]

Measurement model with sensor noise:
\[ y_i^t = g(x^t, u^t) + v_i^t \]
\[ v_i^t \sim N(0, R_i) \]

**Goal:** Robots work together to find
\[ \hat{x}^t = \arg \max_{x^t} p(x^t \mid y_{1:t}^t) \]

Maximum A Posteriori Objective Function

\[ \hat{x}^t = \arg \max_{x^t} p(x^t \mid y_{1:t}^1) \]

Separated Objective Function:

\[ J_i(x_i^t, x_{i-1}^t) = \frac{1}{n} \| x_i^t - f(x_{i-1}^t) \|^2_{Q_t^{-1}} + \| y_i^t - g(x_i^t) \|^2_{R_i^{-1}} + \| x_i^{t-1} - \hat{x}_i^{t-1} \|^2_{P_i^{-1}} \]

Iterations by Agent i:

\[ \hat{x}_i^{(k+1)} = \arg \min_{x_i} \left\{ J_i(x_i) + x_i^T p_i^{(k+1)} + \rho \sum_{j \in \mathcal{N}_{i,t}} \| x_i - \frac{1}{2} \left( \hat{x}_i^{(k)} + \hat{x}_j^{(k)} \right) \|^2 \right\} \]

\[ p_i^{(k+1)} = p_i^{(k)} + \rho \sum_{j \in \mathcal{N}_{i,t}} \left( \hat{x}_i^{(k)} - \hat{x}_j^{(k)} \right) \]
We compare the performance of the DRWT method in initialization for each time step, and perform DRWT with

**Algorithm 1**

1. \[ \hat{x}_i^{(t)} \]
2. \[ \hat{p}_i^{(t)} \]
3. \[ \hat{H}_i^{(t)} \]
4. \[ \hat{Q}_i^{(t)} \]
5. \[ \hat{C}_i^{(t)} \]

for \( i \) in 1 to \( N \)

1. \[ \hat{x}_i^{(t)} = \hat{x}_i^{(t-1)} + \hat{u}_i^{(t)} \]
2. \[ \hat{p}_i^{(t)} = \hat{p}_i^{(t-1)} + \hat{Q}_i^{(t)} \]
3. \[ \hat{H}_i^{(t)} = \hat{H}_i^{(t-1)} + \hat{H}_i^{(t)} \]
4. \[ \hat{Q}_i^{(t)} = \hat{Q}_i^{(t-1)} + \hat{Q}_i^{(t)} \]
5. \[ \hat{C}_i^{(t)} = \hat{C}_i^{(t-1)} + \hat{C}_i^{(t)} \]

end function

**B. CARLA Simulations**

For the simulation trials, each sensor vehicle is equipped with a forward and a backward-facing camera, within CARLA, a simulation test-bed for autonomous driving systems. For the simulation trials, each sensor vehicle is equipped with a forward and a backward-facing camera, within CARLA, a simulation test-bed for autonomous driving systems. We demonstrate our algorithm in a scenario involving multiple vehicles equipped with front facing and rear facing cameras, each with a communication network for a single timestep's estimate. The communication network is equipped with a forward and a backward-facing camera, within CARLA, a simulation test-bed for autonomous driving systems. We demonstrate our algorithm in a scenario involving multiple vehicles equipped with front facing and rear facing cameras, each with a communication network for a single timestep's estimate. The communication network is equipped with a forward and a backward-facing camera, within CARLA, a simulation test-bed for autonomous driving systems.


**100x faster than consensus Kalman Filter**

**Better Estimation Quality than CKF**
vehicles acquire semantic segmentation and depth images at \( 4 \) Hz. The sensing radius of the vehicles is limited to 100m. The relative position of each target vehicle is deduced from the depth and segmentation images and the camera's projection matrix. Each sensor uses its odometry information to transform the relative position of the target into the global coordinate frame corresponding to the measurement used by the vehicle in DRWT. The sensor estimates trajectories of \( T = 5 \) s in length. For this simulation, we assume that the target labeling is known \textit{a priori}. The communication network between sensor vehicles is modeled as a disk graph with a 200 m radius and is updated at 4 Hz. Figure 6 shows the mean squared error of the estimated target trajectories with respect to the centralized trajectory estimate. Collaborative target tracking using DRWT significantly outperforms the estimates made by any single agent. Increasing the number of iterations of DRWT in each estimation round can further reduce the remaining error. Figure 7 shows how the information (represented as the trace of the inverse covariance) corresponding to a given target is apportioned across the network. As the set of sensors tracking a target changes in time, the hand-off procedure enables their joint information to closely match the information of the centralized estimate.

VII. Conclusion

The DRWT algorithm enables a fleet of autonomous vehicles to track other vehicles in urban environment in the presence of occlusions. In this method, each sensor-equipped vehicle estimates the target's state over a rolling window, leading to a scalable algorithm that can be parallelized to multiple targets. We show that DRWT converges to the centralized estimate even with less communication bits per node. Future work will focus on target tracking by vehicles with non-linear dynamics and non-linear sensors such as radar and lidar.

REFERENCES


Distributed Manipulation

Shorinwa and Schwager, IROS 2020.
Distributed Multi-Robot Planning through Contact

The robots approach the rod to manipulate it before sliding the rod along the surface.

Each quadrotor approaches the object to support it before manipulating it to its desired orientation, noticeable from the green cap at the object’s top.
Distributed Optimization for Robots Working Together

Game Theoretic Planning for Robots “Playing” Together

Stanford University
FPV Human Drone Racing

GTP Autonomous Drone Racing

Stanford University
Generalized Nash Style Game

Each Robot has its own objective:

\[
\min_{\theta_i \in \Theta} J_i(\theta_i, \theta_{-i})
\]
\[
\text{s.t. } g(\theta_i, \theta_{-i}) \leq 0
\]

Generalized Nash Equilibrium:

For all robots:

\[
\theta_i^* = \arg \min_{\theta_i \in \Theta} J_i(\theta_i, \theta_{-i}^*)
\]
\[
\text{s.t. } g(\theta_i, \theta_{-i}^*) \leq 0
\]

No robot can unilaterally improve its objective.
Game Theoretic Planner: Iterative Best Response

- Nash Equilibrium is a fixed point of the best response map.
- If IBR converges, it converges to a Nash equilibrium!
- In practice, does not work with constraints (Generalized Nash Games)
Sensitivity Enhanced IBR

Estimate of j’s cost at Nash equilibrium:

\[ \tilde{J}_j^*(\theta_i, \theta_j^k) = J_j(\theta_j^k) + \mu_j^k \frac{\partial g_j}{\partial \theta_i} (\theta_i - \theta_i^k) \]

Collision constraint

Lagrange multiplier

Sensitivity of j’s solution wrt i’s solution

Solve optimization including sensitivity of j to wrt i:

\[ \theta_i^{k+1} = \arg \max_{\theta_i} J_i(\theta_i) - \alpha \left( \mu_j^k \frac{\partial g_j}{\partial \theta_i} \theta_i \right) \]

s.t. “constraints”

20ms per iteration online, solves online in receding horizon

Theorem: Fixed point of iteration equivalent to first order conditions for Nash equilibrium
Nonlinear MPC Iterations

\[
\begin{align*}
\max_{\theta_i} & \quad s_i(p_i^N) \\
\text{s.t.} & \quad p_i^k = p_i^{k-1} + u_i^k & \quad \text{(Dynamics)} \\
& \quad \|p_j^k - p_i^k\| \geq d_{ij}, \quad \forall j \neq i, \quad j \in \{1, \ldots, R\} & \quad \text{(Collision)} \\
& \quad n(p_i^k)^T[p_i^k - \tau(p_i^k)] \leq w_r & \quad \text{(Track)} \\
& \quad \|u_i^k\| \leq \bar{u}_i & \quad \text{(Input)}
\end{align*}
\]

Stanford University
• **Blue car** is using MPC planner (MPC), speed limit: 1.8 m/s
• **Yellow car** is using Game Theoretic Planner (GTP), speed limit: 2.5 m/s
Blocking Scenario

X1:
- Game Theoretic Planner (GTP), speed limit: 5m/s
- Simulated car:
  - MPC planner, speed limit: 6m/s

Visualization of Experiments

Third-person camera view

Stanford University

Stanford University
Zijian Wang, Riccardo Spica and Mac Schwager

all the parameters are omitted since the numbers can be arbitrarily scaled up or down depending on the particular application. The game is fully competitive, meaning that each robot selfishly optimizes its own objective without cooperating with any other robot. All the robots plan their trajectories 3 seconds into the future with 10 planning steps. The maximal number of game iterations is 10 for the game theoretic planner to ensure online planning and execution. We implement our simulation in ROS and C++, with Gurobi as the optimization solver. Utilizing multiple computing cores on an Intel i7-6700HQ CPU, where the simulation is conducted, we run the planners in parallel on 6 different CPU cores.

We run 100 simulations for each of the three scenarios: (a) faster MPC vs slower GTP; (b) faster GTP vs slower MPC; (c) faster GTP and slower GTP. All robots’ initial positions are randomly perturbed around their nominal values shown in Figure 1. Additional randomness also comes from the time delay due to the uncontrollable CPU task scheduling and the fact that the planners are running asynchronously. The robots are required to finish two laps. The statistical results of the simulations are shown in Figure 2, where robot 1–3 always mean the faster robots while robot 4–6 always mean the slower robots. The metric we choose for evaluating the performance is \( \text{lag time} \) (y-axis in the plot), that is, the difference between an agent’s finish time and the winner of the game. Consequently, \( \text{lag time} = 0 \) means the first to finish a particular game.

![Figure 2](http://www.gurobi.com/)

Fig. 2

Statistics of 100 simulations for each of the three scenarios. Robot index 1–3 means faster robots while 4–6 means slower robots. The y-axis indicates the lag time, i.e., the time difference behind the winner of the game. Black dots are the mean lag time. Standard deviation are shown by the bold black vertical bars, and min/max values are represented by the gray bars.

From Figure 2(a), GTP robots win most of the game despite being slower than the MPC robots, verifying the GTP’s ability to block the opponents. The MPC robots do occasionally win the game due to the speed advantage, but overall have higher mean lag time. In comparison, faster GTP in Figure 2(b) outperforms MPC robot by a large margin. More interestingly, Figure 2(c) depicts the game with six GTP robots with different speeds. The result is as expected and the winners are more evenly distributed with smaller lag time differences compared to (a) and (b), presumably because all agents are exploiting Nash equilibrium to avoid being disadvantageous in this case. Some sample snapshots of the simulation process are provided in Figure 1.
Game of Drones
NeurIPS 2019 Competition

AirSim Drone Racing Lab

Madaan et al., PMLR 2020.
Distributed Optimization for Collaborating Robots

Game Theoretic Planning for Competing Robots

Stanford University
Thanks!

Mingyu Wang  Ola Shorinwa  Javier Yu  Trevor Halsted  Alex Koufos  Riccardo Spica

Eric Cristofalo  Zijiang Wang  Edu Montijano  Simon LeCleac’h  Zac Manchester

Stanford University