Lecture #5: Boundary estimation and patrolling algorithm

Francesco Bullo[1] Jorge Cortés[2] **Sonia Martínez**[2]

[1]Department of Mechanical Engineering
University of California, Santa Barbara
bullo@engineering.ucsb.edu

[2]Mechanical and Aerospace Engineering
University of California, San Diego
{cortes,soniamd}@ucsd.edu

Workshop on "Distributed Control of Robotic Networks"
IEEE Conference on Decision and Control
Cancun, December 8, 2008

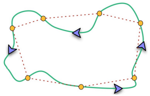
Acknowledgements: Sara Susca, Honeywell

Summary Introduction

- Motion coordination objective: boundary estimation & sensing
- Boundary approximation through polygonal approximations
- Distributed approach based on linear iterations
- Model for "event-driven" robotic networks
- Proof of correctness

Boundary estimation and patrolling algorithms

Objective: Detection/estimation of an evolving 2D boundary



Motivation:

- Demarcation of hazardous environments
- Validation of oceanographic/atmospheric models
  - Delimit areas with abrupt changes of temperature
  - Establish the front of a highly-pollutant expanding substance

Challenges:

- How to devise a decentralized scheme (no fusion center)
- Sudden events require event-driven coordination algorithms

Outline

Event-driven robotic networks

An **event-driven control and communication law** $\mathcal{ECC}$ is defined for a robotic network $\mathcal{S} = (I, \mathcal{R}, E_{\rm cmm})$

Recall that $\mathcal{S} = (I, \mathcal{R}, E_{\rm cmm})$ has physical components

- $I = \{1, \dots, n\}$, the set of UIDs for the robots
- $\mathcal{R} = \{R^{[i]}\} = \{(X, U, X_0, f)\}$, group of mobile robots
- $E_{\rm cmm}$ communication edge map

Here, $x \mapsto (I, E_{\rm cmm})$ topology of communication graph, that is, a **proximity graph**; e.g. a **geometric graph**

Event-driven control and communication law

1. **communication alphabet** — $\mathbb{A}$ including the `null` message
2. **processor state space** — $W^{[i]}$, with initial allowable $W_0^{[i]}$
3. **message-trigger function** — $\text{msg-trig}^{[i]}: X^{[i]} \times W^{[i]} \to \{\texttt{true}, \texttt{false}\}$
4. **message-gen function** — $\text{msg-gen}^{[i]} : X^{[i]} \times W^{[i]} \times I \to \mathbb{A}$
5. **message-recept function** — $\text{msg-rec}^{[i]}: X^{[i]} \times W^{[i]} \times \mathbb{A} \times I \to W^{[i]}$
6. **state-trans trigger functs** — $\text{stf-trig}_k^{[i]}: X^{[i]} \times W^{[i]} \to \{\texttt{true}, \texttt{false}\}$
7. **state-trans functions** — $\text{stf}_k^{[i]} : X^{[i]} \times W^{[i]} \to W^{[i]}$
8. **control function** — $\text{ctl} : X^{[i]} \times W^{[i]} \to U^{[i]}$

Event-driven control and communication law

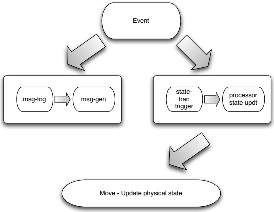
Note the differences between an event-driven and a synchronous CCL:

1. No common time schedule for robots to send/receive msgs or update their states
2. The function msg-gen is replaced by the tripplet (msg-trig, msg-rec, msg-gen)
3. The function stf is replaced by the pairs $(\text{stf-trig}_k, \text{stf}_k)$
4. The control function depends only upon current robot position

Event-driven control and communication law

Executions: Only when an event happens, an agent sends/receives a message or updates its state.

The trigger functs check these constraints acting as **guard maps**



Event-driven robotic networks – formal execution

Evolution of $(\mathcal{S}, \mathcal{ECC})$
with dwell time $\delta \in \mathbb{R}_{\geq 0}$ and $x_0^{[i]} \in X_0^{[i]}$, $w_0^{[i]} \in W_0^{[i]}$, $i \in I$, are $x^{[i]} : \mathbb{R}_{\geq 0} \to X^{[i]}$, and $w^{[i]} : \mathbb{R}_{\geq 0} \to W^{[i]}$, $i \in I$, such that

$$\dot{x}^{[i]}(t) = f\big(x^{[i]}(t), \mathrm{ctl}^{[i]}(x^{[i]}(t), w^{[i]}(t))\big),$$
$$\dot{w}^{[i]}(t) = 0,$$

with $x^{[i]}(0) = x_0^{[i]}$ and $w^{[i]}(0) = w_0^{[i]}$, $i \in I$, and such that

I. for $i \in I$ and $t_1 \in \mathbb{R}_{>0}$, msg generated by i, received by j are

$$y_i^{[j]}(t_1) = \text{msg-gen}^{[i]}\big(x^{[i]}(t_1), w^{[i]}(t_1), j\big),$$
$$w^{[j]}(t_1) = \text{msg-rec}^{[j]}\big(x^{[j]}(t_1), \lim_{t \to t_1^-} w^{[j]}(t), y_i^{[j]}(t_1), i\big),$$

if $\text{msg-trig}^{[i]}(x^{[i]}(t_1), w^{[i]}(t_1)) = \texttt{true}$ and agent i has not transmitted any message during the time interval $]t_1 - \delta, t_1[\cap \mathbb{R}_{>0}$.

Event-driven robotic networks – formal execution

II. for every $i \in I$, $k \in \{1, \dots, K_{\mathrm{stf}}^{[i]}\}$ and $t_2 \in \mathbb{R}_{>0}$ the state-transition function $\mathrm{stf}_k^{[i]}$ is executed, that is,

$$w^{[i]}(t_2) = \mathrm{stf}_k^{[i]}\big(x^{[i]}(t_2), \lim_{t \to t_2^-} w^{[i]}(t)\big),$$

if $\text{stf-trig}_k^{[i]}(x^{[i]}(t_2), w^{[i]}(t_2)) = \texttt{true}$ and there has been no execution of $\mathrm{stf}_k^{[i]}$ during the time interval $]t_2 - \delta, t_2[\cap \mathbb{R}_{>0}$.

Outline

Planar boundary interpolation by inscribed polygons

Given $Q \subseteq \mathbb{R}^2$ a simply connected set or a *body*,
How can we concisely describe ∂Q? use **approximating polygons**
Critical inscribed polygons for convex bodies

- Consider the **symmetric error metric**

$$\delta^S(C, B) = \mu(C \cup B) - \mu(C \cap B),$$

where μ is the Lebesgue measure on $\mathbb{R}^2$

Let $Q_m \subseteq Q$ be an inscribed polygon with vertices $\{q_1, \dots, q_m\}$
Let $s_j \in [0, 2\pi]$ be such that $q_j = \gamma(s_j)$, for $j \in \{1, \dots, m\}$
Then, Q_m is a critical point of δ^S if and only if

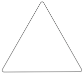
$$\iota(\gamma'(s_j)) \times \iota(q_{j+1} - q_{j-1}) = \mathbf{0}_3, \quad \text{for all } j \in \{1, \dots, m\},$$

where $q_0 = q_m$, $q_{m+1} = q_1$ and $\iota : \mathbb{R}^2 \to \mathbb{R}^3$ is the natural inclusion

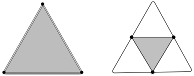
Planar boundary interpolation by inscribed polygons

An illustrative example

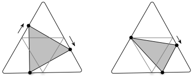
Consider the convex body



Then, critical inscribed polygons are:



But the last critical polygon is a saddle



Planar boundary interpolation by inscribed polygons

- **Asymptotic formula** of McClure and Vitale

Let ρ be curvature radius and κ_{abs} curvature of the boundary
Suppose that ∂Q is of class $\mathcal{C}^2$ with $\kappa_{\text{abs}} > 0$. Then

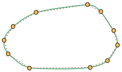
$$\lim_{m\to+\infty} m^2 \delta^S(Q, Q_m^*) = \frac{1}{12}\left(\int_0^{2\pi} \rho(\theta)^{2/3} d\theta\right)$$

Method of empirical distributions

$\gamma_{\text{arc}}(\varrho)$ arc-length parametrization of ∂Q
$\gamma_{\text{polar}}(\theta)$, polar variable parameterization, of ∂Q

Let $q_1, \dots, q_m$ such that $q_i = \gamma_{\text{arc}}(\varrho_i)$ and $q_i = \gamma(\theta_i)$, for $i \in \{1, \dots, m\}$. Then, take the q_i so that

$$\int_{\theta_i}^{\theta_{i+1}} \rho(\theta)^{2/3} d\theta \approx \int_{\theta_{i-1}}^{\theta_i} \rho(\theta)^{2/3} d\theta$$



Planar boundary interpolation by inscribed polygons

Adaptation to **non-convex boundaries** as follows:
Let $q_i = \gamma_{\text{arc}}(\varrho_i)$ and $q_j = \gamma_{\text{arc}}(\varrho_j)$, with $\varrho_i < \varrho_j$, then

$$\mathcal{D}_{\text{curvature}}(q_i, q_j) = \int_{\varrho_i}^{\varrho_j} \kappa_{\text{abs}}(\gamma_{\text{arc}}(\varrho))^{1/3} d\varrho,$$
$$\mathcal{L}(q_i, q_j) = \varrho_j - \varrho_i.$$

are positive only when ∂Q is transversed counterclockwise from q_i to q_j
For $\lambda \in [0, 1]$, define the **pseudo-distance** $\mathcal{D}_\lambda$ between q_i and q_j as

$$\mathcal{D}_\lambda(q_i, q_{i+1}) = \lambda \mathcal{D}_{\text{curvature}}(q_i, q_j) + (1 - \lambda)\mathcal{L}(q_i, q_{i+1}).$$

Interpretation:

$\lambda \approx 1 \implies$ method of empirical distributions
$\lambda \approx 0 \implies$ equal division of boundary
$\lambda \in (0, 1) \implies$ midway approximation

Outline

Network model

From now on, Q is a body with differentiable boundary ∂Q

Robotic Network:
$\mathcal{S}_{\text{bndry}} = (I, \mathcal{R}, E_{\text{cmm}})$, with $I = \{1, \dots, n\}$, where

$$(\partial Q, [-v_{\min}, v_{\max}], \partial Q, (0, \mathbf{e})),$$

- $\mathbf{e}$ vector field tangent to ∂Q (counterclockwise motion)
- E_{cmm} is the ring graph or the Delaunay graph on ∂Q

Assume also that

- unit speed is admissible; i.e. $1 \in [-v_{\min}, v_{\max}]$
- Each robot can sense its own location $p^{[i]} \in \partial Q$, $i \in I$

Boundary estimation tasks

Overall n_{ip} interpolation points are used to approximate ∂Q
Thus, a robot's processor state component is $q^{[i]} \in (\mathbb{R}^2)^{n_{\text{ip}}}$, for $i \in I$

Boundary estimation task
$\mathcal{T}_{\varepsilon\text{-bndry}} : (\partial Q)^n \times ((\mathbb{R}^2)^{n_{\text{ip}}})^n \to \{\texttt{true}, \texttt{false}\}$ for $\mathcal{S}_{\text{bndry}}$ is

$$\mathcal{T}_{\varepsilon\text{-bndry}}(p^{[1]}, \dots, p^{[n]}, q^{[1]}, \dots, q^{[n]}) = \texttt{true} \quad \text{if and only if}$$
$$\left| \mathcal{D}_\lambda(q^{[i]}_{\alpha-1}, q^{[i]}_\alpha) - \mathcal{D}_\lambda(q^{[i]}_\alpha, q^{[i]}_{\alpha+1}) \right| < \varepsilon, \quad \alpha \in \{1, \dots, n_{\text{ip}}\} \text{ and } i \in I.$$

Agent equidistance task $\mathcal{T}_{\varepsilon\text{-eqdstnc}} : (\partial Q)^n \to \{\texttt{true}, \texttt{false}\}$ is $\texttt{true}$

$$\left| \mathcal{L}(p^{[i-1]}, p^{[i]}) - \mathcal{L}(p^{[i]}, p^{[i+1]}) \right| < \varepsilon, \quad \text{for all } i \in I,$$

where $\mathcal{L}$ is the counterclockwise arc-length distance along ∂Q.

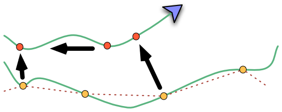
Outline

Estimate update and cyclic balancing law

We will present this event-driven law incrementally

- Single robot estimate update law
- Cooperative estimate update law
- Cyclic balancing algorithm for agent equidistance task

Single robot estimate update law



Describes the single-robot operation "projection-and-update"

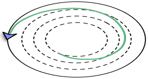
Single robot estimate update law

Processor state of robot $i \in I$

1. a counter $\mathtt{nxt} \in \{1, \dots, n_{\mathrm{ip}}\}$
   index of interpolation point the agent projects next;
2. a boundary representation

   $$\{(q_\alpha, v_\alpha) \in (\mathbb{R}^2)^2 \mid \alpha \in \{1, \dots, n_{\mathrm{ip}}\}\},$$

   here q_α is the position of the α int. point
   and v_α is the tangent vector of ∂Q at q_α
3. a curve $\mathtt{path} : [0, t] \to \mathbb{R}^2$
   followed by the agent until present time t

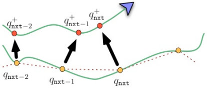


Single robot estimate update law

Rules for processor update – Informal description

- **Rule #1** When and how to project $q_{\mathtt{nxt}}$ onto ∂Q

  Let $q_{\mathtt{nxt}}$ be the point to be projected and $v_{\mathtt{nxt}}$ its tangent
  The projection takes place when the agent crosses the line, denoted by $\mathtt{line}_{\mathtt{nxt}}$, that passes through $q_{\mathtt{nxt}}$ and is perpendicular to $v_{\mathtt{nxt}}$. At this crossing time $q^+_{\mathtt{nxt}} \in \mathtt{path}$, is the point on $\mathtt{path}$ where the agent trajectory crosses the line $\mathtt{line}_{\mathtt{nxt}}$



We write

$$q^+_{\mathtt{nxt}} := \mathtt{perp\text{-}proj}(q_{\mathtt{nxt}}, v_{\mathtt{nxt}}, \mathtt{path}).$$

Single robot estimate update law

- **Rule #2** When and how to optimize the interpolation point

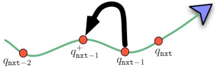
  After the projection of $q_{\mathtt{nxt}}$, $q_{\mathtt{nxt}-1}$ is re-placed to balance its pseudodistance to neighboring interpolation points. Specifically, define $\mathtt{cyclic\text{-}balance} : (\mathbb{R}^2)^3 \times \mathcal{C}(\mathbb{R}^2) \to \mathbb{R}^2$ by

$$\mathtt{cyclic\text{-}balance}(q_{\mathtt{nxt}-2}, q_{\mathtt{nxt}-1}, q_{\mathtt{nxt}}, \mathtt{path}) = q^* \text{ s.t.}$$

$$\mathcal{D}_\lambda(q_{\mathtt{nxt}-2}, q^*) = \frac{3}{4}\mathcal{D}_\lambda(q_{\mathtt{nxt}-2}, q_{\mathtt{nxt}-1}) + \frac{1}{4}\mathcal{D}_\lambda(q_{\mathtt{nxt}-1}, q_{\mathtt{nxt}})$$

In this way,

$$q^+_{\mathtt{nxt}-1} := \mathtt{cyclic\text{-}balance}(q_{\mathtt{nxt}-2}, q_{\mathtt{nxt}-1}, q_{\mathtt{nxt}}, \mathtt{path})$$



Single robot estimate update law

The optimal placement $q^+_{\mathtt{nxt}-1}$ can be equivalently defined by

$$\mathcal{D}_\lambda(q^+_{\mathtt{nxt}-1}, q_{\mathtt{nxt}}) = \frac{1}{4}\mathcal{D}_\lambda(q_{\mathtt{nxt}-2}, q_{\mathtt{nxt}-1}) + \frac{3}{4}\mathcal{D}_\lambda(q_{\mathtt{nxt}-1}, q_{\mathtt{nxt}}),$$

so that it achieves the balancing property that

$$\begin{bmatrix} \mathcal{D}_\lambda(q_{\mathtt{nxt}-2}, q^+_{\mathtt{nxt}-1}) \\ \mathcal{D}_\lambda(q^+_{\mathtt{nxt}-1}, q_{\mathtt{nxt}}) \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}\begin{bmatrix} \mathcal{D}_\lambda(q_{\mathtt{nxt}-2}, q_{\mathtt{nxt}-1}) \\ \mathcal{D}_\lambda(q_{\mathtt{nxt}-1}, q_{\mathtt{nxt}}) \end{bmatrix}$$

Finally, a last useful operation

$$v := \mathtt{tangentat}(\mathtt{path}, q).$$

Single robot estimate update law

Formal Single-Robot Estimate Update Law description

`Robot:` single robot moving at constant speed along ∂Q, continuously recording its trajectory
`Event-driven Algorithm:` Single-Robot Estimate Update Law
`Processor State:` $w = (\texttt{nxt}, \{(q_\alpha, v_\alpha)\}_{\alpha=1}^{n_{\mathrm{ip}}}, \texttt{path})$, where

`nxt`	$\in \{1, \dots, n_{\mathrm{ip}}\}$,	initially equal to index of the interpolation point closest to robot moving counterclockwise
$\{(q_\alpha, v_\alpha)\}_{\alpha=1}^{n_{\mathrm{ip}}}$	$\subset \mathbb{R}^2 \times \mathbb{R}^2$,	initially counterclockwise along boundary
`path`	$\in \mathcal{C}(\mathbb{R}^2)$,	continuously recording agent's trajectory

Single robot estimate update law

% A state transition is triggered when the agent crosses a certain line
`function` stf-trig(p, w)

```
line_nxt := line through point q_nxt perpendicular to direction v_nxt
if p ∈ line_nxt then
    return true
else
    return false
```

% The current interpolation point and tangent vector are projected and the previous interpolation point is optimized along the new boundary
`function` stf(p, w)

$\{(q_\alpha^+, v_\alpha^+)\}_{\alpha=1}^{n_{\mathrm{ip}}} := \{(q_\alpha, v_\alpha)\}_{\alpha=1}^{n_{\mathrm{ip}}}$
$q_{\texttt{nxt}}^+ := \texttt{perp-proj}(q_{\texttt{nxt}}, v_{\texttt{nxt}}, \texttt{path})$
$q_{\texttt{nxt}-1}^+ := \texttt{cyclic-balance}(q_{\texttt{nxt}-2}, q_{\texttt{nxt}-1}, q_{\texttt{nxt}}^+, \texttt{path})$
$v_{\texttt{nxt}}^+ := \texttt{tangentat}(\texttt{path}, q_{\texttt{nxt}}^+)$
$v_{\texttt{nxt}-1}^+ := \texttt{tangentat}(\texttt{path}, q_{\texttt{nxt}-1}^+)$
return $(\texttt{nxt}+1, \{(q_\alpha^+, v_\alpha^+)\}_{\alpha=1}^{n_{\mathrm{ip}}}, \texttt{path})$

Outline

Cooperative estimate update law

Informal description

Parallel version of the Single-Robot Estimate Update Law:

- each agent updates its boundary representation separately
- every time an agent updates two interpolation points, this agent transmits these to its neighbors
- In turn, the neighbors record the updates in their individual boundary representation

Agents must satisfy the **two-hop separation rule**

> A group of $n \geq 2$ agents is
> two-hop separated along the interpolation points
> if $\texttt{nxt}^{[i-1]} \leq \texttt{nxt}^{[i]} - 2$ for all $i \in I$ at all times

Cooperative estimate update law

Formal description of the cooperative estimate update law

`Robotic Network:` $\mathcal{S}_{\text{bndry}}$, assume agents
with absolute sensing of own position, communicating
with clockwise and counterclockwise neighbors

`Event-driven Algorithm:` COOPERATIVE ESTIMATE UPDATE LAW
`Alphabet:` $\mathbb{A} = \{1, \dots, n_{\text{ip}}\} \times (\mathbb{R}^2)^2 \times (\mathbb{R}^2)^2 \cup \{\texttt{null}\}$
`Processor State`, `function` stf-trig, and `function` stf

same as in SINGLE-ROBOT ESTIMATE UPDATE LAW

Cooperative estimate update law

%A transmission is triggered right after the interpolation points are updated
`function` msg-trig(p, w)
return stf-trig(p, w)

%The updated interpolation points (and reference label) are transmitted
`function` msg-gen(p, w, i)
return $(\texttt{nxt}, (q_{\texttt{nxt}-1}, v_{\texttt{nxt}-1}), (q_{\texttt{nxt}-2}, v_{\texttt{nxt}-2}))$

%The received updated interpolation points are stored
`function` msg-rec(p, w, y, i)
$\{(q^+_\alpha, v^+_\alpha)\}_{\alpha=1}^{n_{\text{ip}}} := \{(q_\alpha, v_\alpha)\}_{\alpha=1}^{n_{\text{ip}}}$
$(\texttt{nxtrec}, y_1, y_2) := y$
$(q^+_{\texttt{nxtrec}-1}, v^+_{\texttt{nxtrec}-1}) := y_1$
$(q^+_{\texttt{nxtrec}-2}, v^+_{\texttt{nxtrec}-2}) := y_2$
return $(\texttt{nxt}, \{(q^+_\alpha, v^+_\alpha)\}_{\alpha=1}^{n_{\text{ip}}}, \texttt{path})$

Outline

Cyclic balancing algorithm for agent equidistance law

Motion control law for each agent
Suppose agent $i \in I$ is at position $p^{[i]}$ moving in continuous time with speed $v^{[i]}$ along ∂Q. Then,

$$v^{[i]} = 1 + k_{\text{prop}}\big(\mathcal{L}(p^{[i]}, p^{[i+1]}) - \mathcal{L}(p^{[i-1]}, p^{[i]})\big),$$

where $k_{\text{prop}} \in \mathbb{R}_{>0}$ is a fixed control gain.

To enforce the constraint $v \in [-v_{\min}, v_{\max}]$, we introduce

$$v^{[i]} = \mathsf{sat}_{[v_{\min}, v_{\max}]}\left(1 + k_{\text{prop}}\big(\mathcal{L}(p^{[i]}, p^{[i+1]}) - \mathcal{L}(p^{[i-1]}, p^{[i]})\big)\right),$$

where

$$\mathsf{sat}_{[a,b]}(x) = \begin{cases} a, & \text{if } x < a, \\ x, & \text{if } x \in [a, b], \\ b, & \text{if } x > b. \end{cases}$$

Cyclic balancing algorithm for agent equidistance law

Approximation of
the counterclockwise arc-length distance between robots

Given $\{q_1, \dots, q_{n_{\mathrm{ip}}}\}$ and $r_1, r_2 \in \partial Q$, let the indices $[r_1]$ and $[r_2]$ of the counterclockwise-closest interpolation points from r_1, r_2, respectively.

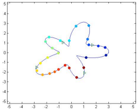
$$\mathcal{L}(r_1, r_2) = \mathcal{L}(r_1, q_{[r_1]}) + \sum_{\alpha=[r_1]}^{[r_2]-2} \mathcal{L}(q_\alpha, q_{\alpha+1}) + \mathcal{L}(q_{[r_2]-1}, r_2) \quad (1)$$

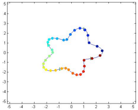
$$\approx \mathsf{dist}_2(r_1, q_{[r_1]}) + \sum_{\alpha=[r_1]}^{[r_2]-2} \mathsf{dist}_2(q_\alpha, q_{\alpha+1}) + \mathsf{dist}_2(q_{[r_2]-1}, r_2). \quad (2)$$

Both (1) or (2) approximations may be possible, depending on info available

Outline

Simulations





Outline

Correctness of the estimate updt and cyclic blncg law

Theorem (Correctness of the exact and approximate laws)

On the network $\mathcal{S}_{\text{bndry}}$, along evolutions with the two-hop property,

1. *the* ESTIMATE UPDATE AND BALANCING LAW *achieves the boundary estimation task $\mathcal{T}_{\varepsilon\text{-bndry}}$ and the agent equidistance task $\mathcal{T}_{\varepsilon\text{-eqdstnc}}$ for any $\varepsilon \in \mathbb{R}_{>0}$ if the boundary is time-independent, and*
2. *the* APPROXIMATE ESTIMATE AND BALANCING LAW *achieves the boundary estimation task $\mathcal{T}_{\varepsilon\text{-bndry}}$ and the agent equidistance task $\mathcal{T}_{\varepsilon\text{-eqdstnc}}$ for some $\varepsilon \in \mathbb{R}_{>0}$ if the boundary varies in a continuously differentiable way and sufficiently slowly with time, and its length is upper bounded.*

Proofs – Time-invariant boundary

Let us focus on $\mathcal{T}_{\varepsilon\text{-bndry}}$ in two cases:

- Time-invariant boundaries and no approximations
- Time-varying boundaries and approximations

That is, we need to show that

$$\left|\mathcal{D}_\lambda(q^{[i]}_{\alpha-1}, q^{[i]}_\alpha) - \mathcal{D}_\lambda(q^{[i]}_\alpha, q^{[i]}_{\alpha+1})\right| < \varepsilon,$$

For **time-invariant** boundaries,

suppose that `path` is exact and there are no other approximations

Consider $\mathcal{D}_\alpha = \mathcal{D}_\lambda(q_\alpha, q_{\alpha+1})$, $\alpha \in \{1, \dots, n_{\text{ip}}\}$, and $\mathcal{D} = (\mathcal{D}_1, \dots, \mathcal{D}_{n_{\text{ip}}}) \in \mathbb{R}^{n_{\text{ip}}}_{>0}$. A single robot changes $\mathcal{D}$ as

$$\begin{bmatrix} \mathcal{D}_{\texttt{nxt}-2} \\ \mathcal{D}_{\texttt{nxt}-1} \end{bmatrix}^+ = \frac{1}{4}\begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}\begin{bmatrix} \mathcal{D}_{\texttt{nxt}-2} \\ \mathcal{D}_{\texttt{nxt}-1} \end{bmatrix}$$

Proofs – Time-invariant boundary

Denote $\mathcal{D}^+_\alpha = \mathcal{D}_\lambda(q^+_\alpha, q^+_{\alpha+1})$, for $\alpha \in \{1, \dots, n_{\text{ip}}\}$
For $\texttt{nxt} \in \{1, \dots, n_{\text{ip}}\}$, define $A_{\texttt{nxt}} \in \mathbb{R}^{n_{\text{ip}} \times n_{\text{ip}}}$ by

$$(A_{\texttt{nxt}})_{jk} = \begin{cases} 3/4, & \text{if } (j,k) \text{ equals } (\texttt{nxt}-1, \texttt{nxt}-1) \text{ or } (\texttt{nxt}-2, \texttt{nxt}-2), \\ 1/4, & \text{if } (j,k) \text{ or } (j,k) \text{ equals } (\texttt{nxt}-1, \texttt{nxt}-2), \\ \delta_{jk}, & \text{otherwise} \end{cases}$$

Define a graph $G_{\texttt{nxt}}$

over $\{1, \dots, n_{\text{ip}}\}$, with the single edge $(\texttt{nxt}-1, \texttt{nxt}-2)$

In summary, the matrix $A_{\texttt{nxt}}$ determines the change of state $\mathcal{D}^+ = A_{\texttt{nxt}}\mathcal{D}$ when a projection-and-placement event takes place with counter `nxt` and its associated graph is $G_{\texttt{nxt}}$.

Proofs – Time-invariant boundary

Let ℓ be the time marking a projection-and-placement event. Then,

$$\mathcal{D}(\ell) = A_{\texttt{nxt}(\ell)}\mathcal{D}(\ell-1), \qquad \ell \geq 1$$

Observe that:

- $A_{\texttt{nxt}(\ell)}$ is non-degenerate, symmetric and doubly stochastic
- $\cup_{\tau \geq \ell} G_{\texttt{nxt}(\tau)}$ is connected

Thus,

$$\lim_{\ell\to\infty} \mathcal{D}_\alpha(\ell) = \frac{1}{n_{\text{ip}}}\sum_{\alpha=1}^{n_{\text{ip}}} \mathcal{D}_\alpha(0) = \frac{1}{n_{\text{ip}}}\text{length}(\partial Q)$$

Reasoning can be extended to multiple agents with two hop condition

Proofs – Time-varying boundary

Consider now **time-varying boundaries**, $t \mapsto \partial Q(t)$, such that:

- $\mathsf{length}(\partial Q(t))$ is uniformly bounded for all t
- The boundary $\partial Q(t)$ is smooth for all t
- No other approximations take place; e.g. no distance approx

Observe that now:

- The state trajectory $\mathcal{D} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{n_{\text{ip}}}$ will be time-varying
- We redefine $\mathcal{D}_\alpha$ to measure pseudodistance between interpolation points along `path`

The bound on $\mathsf{length}(\partial Q)$ guarantees finite time between two updates

Proofs – Time-varying boundary

Let ℓ mark the projection-and-update event times
We model the time-varying boundary effect as

$$\mathcal{D}(\ell) = A_{\texttt{nxt}(\ell)}\big(\mathcal{D}(\ell-1) + \mathcal{U}(\ell)\big),$$

here $\mathcal{U}$ is a disturbance such that:

- Its non-zero entries are the $(\texttt{nxt}(\ell)-1)^{\text{th}}$ and $(\texttt{nxt}(\ell)-2)^{\text{th}}$
- it vanishes at the rate of change of the boundary

Now define the disagreement vector $\ell \to \mathbf{d}(\ell) \in \mathsf{span}\{\mathbf{1}_{n_{\text{ip}}}\}^{\perp}$ by

$$\mathbf{d}(\ell) = \mathcal{D}(\ell) - \frac{\mathbf{1}_{n_{\text{ip}}}^T \mathcal{D}(\ell)}{n_{\text{ip}}} \mathbf{1}_{n_{\text{ip}}}$$

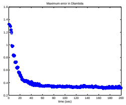
Proofs – Time-varying boundary

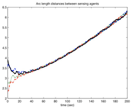
Since $A_{\texttt{nxt}(\ell)}$ is doubly stochastic, the update law for $\mathbf{d}$ is

$$\mathbf{d}(\ell) = A_{\texttt{nxt}(\ell)}\mathbf{d}(\ell-1) + \mathbf{u}(\ell), \quad \ell \in \mathbb{N},$$

where $\mathbf{u}(\ell) = \mathcal{U}(\ell) - \frac{1}{n_{\text{ip}}}\mathbf{1}_{n_{\text{ip}}}^T \mathcal{U}(\ell)\mathbf{1}_{n_{\text{ip}}}$
(correct for one or several robots with two-hop property)

Our task is satisfied if equivalently $\mathbf{d} \approx 0$. In simulation,





Proofs – Time-varying boundary

A subsequence of $\mathbf{d}(\ell)$ will help us conclude the result

Define $\ell_k \in \mathbb{N}$, for $k \in \mathbb{N}$,

- $\ell_1 = 1$, and assume agent 1 executes the first projection-and-placement event with index $\texttt{nxt}(1)$,
- $\ell_k \geq 2$ be the k-th time when agent 1 does the projection-and-placement event with the same index $\texttt{nxt}(1)$

It can be seen that $\ell_k - \ell_{k-1} \leq 2n \cdot n_{\text{ip}}$.

Define now $\mathcal{A}_{\ell_k} \in \mathbb{R}^{n_{\text{ip}} \times n_{\text{ip}}}$, for $k \in \mathbb{N}$, by $\mathcal{A}(1) = A_{\texttt{nxt}(1)}$. Then

$$\mathcal{A}(\ell_k) = A_{\texttt{nxt}(\ell_k)} \cdots A_{\texttt{nxt}(\ell_{k-1}+2)} A_{\texttt{nxt}(\ell_{k-1}+1)}, \quad \text{for } k \geq 2,$$

where $\mathcal{A}(\ell_k)$ is doubly stochastic and irreducible

Proofs – Time-varying boundary

Now we can rewritte

$$\mathbf{d}(\ell_k) = \mathcal{A}(\ell_k)\mathbf{d}(\ell_{k-1}) + \sum_{\ell=\ell_{k-1}+1}^{\ell_k} A_{\mathtt{nxt}(\ell_k)} \cdots A_{\mathtt{nxt}(\ell+1)}\mathbf{u}(\ell)$$
$$= \mathcal{A}(\ell_k)\mathbf{d}(\ell_{k-1}) + \mathcal{B}(\ell_k)\mathbf{u}_{\text{stacked}}(\ell_k),$$

where $\mathbf{u}_{\text{stacked}}(\ell_k)$ contains all vectors $\mathbf{u}(\ell_{k-1}+1), \dots, \mathbf{u}(\ell_k)$

Let's prove the system is Input-to-State (ISS) stable

Candidate ISS-Lyapunov function for $x(\ell+1) = f(x(\ell), u(\ell))$ is

- V is continuously differentiable
- $\exists\, \alpha_1, \alpha_2 \in K_\infty$: $\alpha_1(\|x\|_2) \le V(x) \le \alpha_2(\|x\|_2)$
- $\exists\, \alpha_3 \in K_\infty$ and $\sigma \in K$ with

$$V(f(x,u)) - V(x) \le -\alpha_3(\|x\|_2) + \sigma(\|u\|_2)$$

Proofs – Time-varying boundary

Take $V : \mathbb{R}^{n_{\text{ip}}} \to \mathbb{R}_{\ge 0}$ by $V(x) = x^T x$
V verifies the first two conditions to be and ISS-Lyapunov function

For the last one, compute

$$V(\mathbf{d}(\ell_{k+1})) - V(\mathbf{d}(\ell_k)) = -\mathbf{d}(\ell_k)^T R(\ell_k)\mathbf{d}(\ell_k)$$
$$+ \mathbf{u}_{\text{stacked}}^T(\ell_k)\mathbf{u}_{\text{stacked}}(\ell_k) + 2\mathbf{u}_{\text{stacked}}^T(\ell_k)\mathcal{A}(\ell_k)\mathbf{d}(\ell_k),$$

Here, $R(\ell_k) = I_{n_{\text{ip}}} - \mathcal{A}(\ell_k)^T\mathcal{A}(\ell_k)$ is positive semidefinite,
0 is a simple eigenvalue associated with $\mathbf{1}_{n_{\text{ip}}}$

In this way, $-x^T R(\ell_k)x$ is strictly negative for all $x \notin \mathsf{span}\{\mathbf{1}_{n_{\text{ip}}}\}^\perp$

Proofs – Time-varying boundary

Let $\mathcal{A}_s$ be any of the $\mathcal{A}(\ell_k)$
Define the set of nonzero eigenvalues of $\mathcal{A}_s$ by

$$S_s = \{\lambda \in \mathbb{R} \mid \det\left(\lambda I_{n_{\text{ip}}} - (\mathcal{A}_s^T\mathcal{A}_s - I_{n_{\text{ip}}})\right) = 0\} \setminus \{0\}$$

and take $\bar{r} = \min_s \min\{|\lambda| \mid \lambda \in S_s\}$. Note that $\bar{r} > 0$.
We can then write

$$V(\mathbf{d}(\ell_k+1)) - V(\mathbf{d}(\ell_k)) \le -\alpha_3(\|\mathbf{d}(\ell_k)\|) + \sigma(\|\mathbf{u}_{\text{stacked}}(\ell_k)\|),$$

where $\alpha_3(\|\mathbf{d}\|) = \frac{1}{2}\bar{r}\|\mathbf{d}\|^2$ and $\sigma(\|\mathbf{u}_{\text{stacked}}\|) = (\frac{2}{\bar{r}}+1)\|\mathbf{u}_{\text{stacked}}\|^2$.

Thus the system defined on $\mathbf{d}(\ell)$ is input-to-state stable
The input-to-state stability implies $\exists\, \varepsilon > 0$ so that $\mathcal{T}_{\varepsilon\text{-bndry}}$ is satisfied

Summary

This chapter presents a detailed treatment of a boundary estimation problem based on:

- A new model of event-driven robotic network
- Linear interpolation theory
- Consensus and ISS stability theory

Comments:

- Possible extensions to e.g. include model of changing boundary
- Inner-outer algorithms leading to critical inscribed/circumscribed polygons
- Boundary patrolling with “bouncing beads”
- Similar interpolation ideas for distributed “field estimation”

Bibliographical Notes

On hybrid systems:

- A. J. van der Schaft and H. Schumacher. *An Introduction to Hybrid Dynamical Systems*, volume 251 of *Lecture Notes in Control and Information Sciences*. Springer, 2000

On interpolated curves by polygons:

- P. M. Gruber. Approximation of convex bodies. In P. M. Gruber and J. M. Willis, editors, *Convexity and its Applications*, pages 131--162. Birkhäuser, 1983

Bibliographical Notes

Boundary estimation and tracking:

- D. Marthaler and A. L. Bertozzi. Tracking environmental level sets with autonomous vehicles. In S. Butenko, R. Murphey, and P. M. Pardalos, editors, *Recent Developments in Cooperative Control and Optimization*, pages 317--330. Kluwer Academic Publishers, 2003
- D. W. Casbeer, S.-M. Li, R. W. Beard, R. K. Mehra, and T. W. McLain. Forest fire monitoring with multiple small UAVs. In *American Control Conference*, pages 3530--3535, Portland, OR, June 2005
- D. W. Casbeer, D. B. Kingston, R. W. Beard, T. W. Mclain, S.-M. Li, and R. Mehra. Cooperative forest fire surveillance using a team of small unmanned air vehicles. *International Journal of Systems Sciences*, 37(6):351--360, 2006
- F. Zhang and N. E. Leonard. Generating contour plots using multiple sensor platforms. In *IEEE Swarm Intelligence Symposium*, pages 309--316, Pasadena, CA, June 2005