Optimizing Synchronization Times for Distributed Tracking of a Mobile Asset in GPS-denied Environments

Caleb M. Bowyer and John M. Shea

University of Florida, ECE

Big Picture

- Desire low cost, low complexity, robust, high-performance solutions to tracking/RADAR in GPS-denied environments

Big Picture

- Desire low cost, low complexity, robust, high-performance solutions to tracking/RADAR in GPS-denied environments
  - Low cost, low complexity $\Rightarrow$ small sensors with unreliable clocks

Big Picture

- Desire low cost, low complexity, robust, high-performance solutions to tracking/RADAR in GPS-denied environments
  - Low cost, low complexity $\Rightarrow$ small sensors with unreliable clocks
  - Robust: no single point of failure $\Rightarrow$ distributed sensors with robustness to failure of individual sensors

Big Picture

- Desire low cost, low complexity, robust, high-performance solutions to tracking/RADAR in GPS-denied environments
  - Low cost, low complexity $\Rightarrow$ small sensors with unreliable clocks
  - Robust: no single point of failure $\Rightarrow$ distributed sensors with robustness to failure of individual sensors
  - High-performance $\Rightarrow$ produce reliable localization estimates using noisy measurements and noisy clocks

Big Picture

- Desire low cost, low complexity, robust, high-performance solutions to tracking/RADAR in GPS-denied environments
  - Low cost, low complexity $\Rightarrow$ small sensors with unreliable clocks
  - Robust: no single point of failure $\Rightarrow$ distributed sensors with robustness to failure of individual sensors
  - High-performance $\Rightarrow$ produce reliable localization estimates using noisy measurements and noisy clocks
- Given accurate locations and tightly synchronized clocks, distributed sensor networks can produce accurate location estimates

Big Picture

- Desire low cost, low complexity, robust, high-performance solutions to tracking/RADAR in GPS-denied environments
  - Low cost, low complexity $\Rightarrow$ small sensors with unreliable clocks
  - Robust: no single point of failure $\Rightarrow$ distributed sensors with robustness to failure of individual sensors
  - High-performance $\Rightarrow$ produce reliable localization estimates using noisy measurements and noisy clocks
- Given accurate locations and tightly synchronized clocks, distributed sensor networks can produce accurate location estimates
- Clock synchronization requires communication among sensors and localization may not be possible during the synchronization times

Big Picture

- Desire low cost, low complexity, robust, high-performance solutions to tracking/RADAR in GPS-denied environments
  - Low cost, low complexity $\Rightarrow$ small sensors with unreliable clocks
  - Robust: no single point of failure $\Rightarrow$ distributed sensors with robustness to failure of individual sensors
  - High-performance $\Rightarrow$ produce reliable localization estimates using noisy measurements and noisy clocks
- Given accurate locations and tightly synchronized clocks, distributed sensor networks can produce accurate location estimates
- Clock synchronization requires communication among sensors and localization may not be possible during the synchronization times
- Need to optimize between localization and synchronization to maximize performance

System Model 1

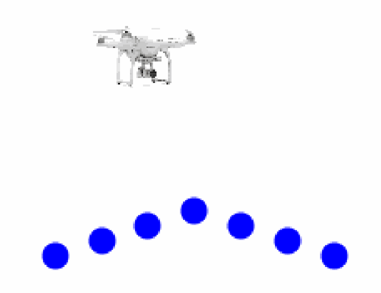
- Fixed network of m sensing agents

System Model 1

- Fixed network of m sensing agents
- Single asset to be tracked:
  - Asset transmits beacon signal at known times to agents to facilitate tracking in GPS-denied environment
  - Asset moves according to known Markov model

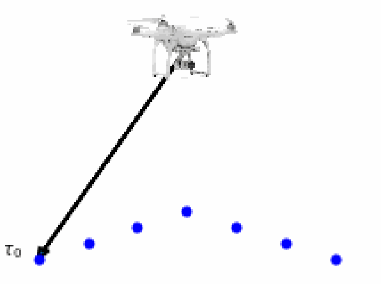
System Model 1

- Fixed network of m sensing agents
- Single asset to be tracked:
  - Asset transmits beacon signal at known times to agents to facilitate tracking in GPS-denied environment
  - Asset moves according to known Markov model



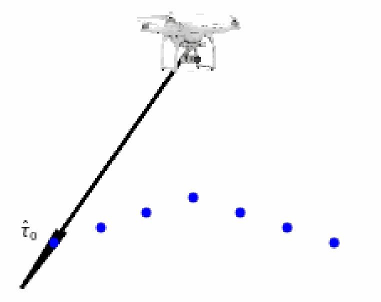
System Model 2

- Sensors measure time-of-flights (ToFs) of beacon signal and localizes (LOC) asset by fusing these measurements



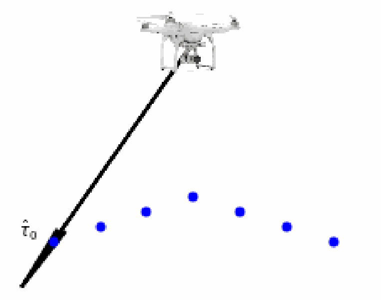
System Model 3

- Each agent's clock drifts independently and variance of clock signals increase with time



System Model 3

- Each agent's clock drifts independently and variance of clock signals increase with time



- Agents can synchronize (SYNCH) clocks at expense of not being able to measure ToFs during that time

Model-Free Localization

- Let coordinates of asset and sensor i in interval k be $(x_{k,a}, y_{k,a}, z_{k,a})$ and (x_i, y_i, z_i)

Model-Free Localization

- Let coordinates of asset and sensor i in interval k be $(x_{k,a}, y_{k,a}, z_{k,a})$ and (x_i, y_i, z_i)
- Using sensor $m-1$ as a reference, form linear equations $\mathbf{A} \cdot \mathbf{v}_k = \boldsymbol{\beta}_k$
- Here $\mathbf{v}_k = \left[x_{k,a}, y_{k,a}, z_{k,a}\right]^T$, $\mathbf{A}$ is a matrix with row i given by

$$\mathbf{A}_i = \left[2(x_i - x_{m-1}),\ 2(y_i - y_{m-1}),\ 2(z_i - z_{m-1})\right], \quad i \in \{0, 1, \ldots, m-2\},$$

and $\boldsymbol{\beta}_k$ is a column vector with component

$$\beta_i = c^2\left(\widehat{\tau}_{k,i}^2 - \widehat{\tau}_{k,m-1}^2\right) - \left(x_i^2 - x_{m-1}^2\right) - \left(y_i^2 - y_{m-1}^2\right) - \left(z_i^2 - z_{m-1}^2\right), \qquad i \in \{0, 1, \ldots, m-2\}$$

Localization Solution

- The least squares solution is given by $[\widehat{x}_{k,a}, \widehat{y}_{k,a}, \widehat{z}_{k,a}]^T = \mathbf{A}^\dagger \boldsymbol{\beta}_k$ where $\mathbf{A}^\dagger = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T$ is the Moore-Penrose pseduo-inverse of $\mathbf{A}$

Improving Localization and Coordinating Synchronization

- Pure localization generally not good enough because of noisy clocks
- Does not inform system of when SYNC is needed
- Resolve both problems by treating tracking problem as HMM and treating choice of SYNC/LOC as control problem

Improving Localization and Coordinating Synchronization

- Pure localization generally not good enough because of noisy clocks
- Does not inform system of when SYNC is needed
- Resolve both problems by treating tracking problem as HMM and treating choice of SYNC/LOC as control problem
- Since true state of asset never known, result is Partially Observable Markov Decision Process (POMDP)

POMDP General Form

1. A finite set of states $\mathcal{X}$

POMDP General Form

1. A finite set of states $\mathcal{X}$
2. A finite set of controls $\mathcal{U}$

POMDP General Form

1. A finite set of states $\mathcal{X}$
2. A finite set of controls $\mathcal{U}$
3. A continuous set of observations $\mathcal{Z}$

POMDP General Form

1. A finite set of states $\mathcal{X}$
2. A finite set of controls $\mathcal{U}$
3. A continuous set of observations $\mathcal{Z}$
4. A state-to-state transition function:
   $p_{ij}(u) = \Pr(X_{k+1} = j | X_k = i, U_k = u)$

POMDP General Form

1. A finite set of states $\mathcal{X}$
2. A finite set of controls $\mathcal{U}$
3. A continuous set of observations $\mathcal{Z}$
4. A state-to-state transition function:
   $p_{ij}(u) = \Pr(X_{k+1} = j | X_k = i, U_k = u)$
5. A state-to-observation transition function:
   $q_{jz}(u) = f(Z_{k+1} = z | X_{k+1} = j, U_k = u)$, and

POMDP General Form

1. A finite set of states $\mathcal{X}$
2. A finite set of controls $\mathcal{U}$
3. A continuous set of observations $\mathcal{Z}$
4. A state-to-state transition function:
   $p_{ij}(u) = \Pr(X_{k+1} = j | X_k = i, U_k = u)$
5. A state-to-observation transition function:
   $q_{jz}(u) = f(Z_{k+1} = z | X_{k+1} = j, U_k = u)$, and
6. A cost function
   $c(x, u, z)$

Control Set

- Controls: $\mathcal{U} = \{u_l, u_s\}$

Control Set

- Controls: $\mathcal{U} = \{u_l, u_s\}$
  - u_l: localize (**loc**)

Control Set

- Controls: $\mathcal{U} = \{u_l, u_s\}$
  - u_l: localize (**loc**)
  - u_s: synchronize (**synch**)

State Space

- $X_k = (M_k, T_k^{(s)})$:

State Space

- $X_k = (M_k, T_k^{(s)})$:
  - M_k is the state of the asset's movement

State Space

- $X_k = (M_k, T_k^{(s)})$:
  - M_k is the state of the asset's movement
  - $T_k^{(s)}$ is the number of time slots since last **sync**

State Space

- $X_k = (M_k, T_k^{(s)})$:
  - M_k is the state of the asset's movement
  - $T_k^{(s)}$ is the number of time slots since last **sync**

  and
- Note that at time k, $T_k^{(s)}$ is known (deterministic) given the previous controls $u_0, u_1, \ldots, u_{k-1}$

Belief States, Observation Sequences and Control Sequences

- Given:
  - $\mathbf{z}_k$: vector of observations up to interval k
  - $\mathbf{u}_{k-1}$: vector of controls leading up to interval $k-1$

Belief States, Observation Sequences and Control Sequences

- Given:
  - $\mathbf{z}_k$: vector of observations up to interval k
  - $\mathbf{u}_{k-1}$: vector of controls leading up to interval $k-1$
- Belief state at interval k is $\mathbf{b}_k$:

$$b_k(x) = \Pr\left(X_k = x \middle| \mathbf{z}_k, \mathbf{u}_{k-1}\right)$$

Belief Update

- Continuous observation space (localization results) – most papers consider finite observation space

$$b_{k+1}(x_{k+1}) = \frac{f(\mathbf{z}_{k+1}, x_{k+1} \,|\mathbf{u}_k\,)}{f(\mathbf{z}_{k+1}\,|\mathbf{u}_k\,)}, \text{ where}$$

Belief Update

- Continuous observation space (localization results) – most papers consider finite observation space

$$b_{k+1}(x_{k+1}) = \frac{f(\mathbf{z}_{k+1}, x_{k+1} \,|\mathbf{u}_k)}{f(\mathbf{z}_{k+1} \,|\mathbf{u}_k)}, \text{ where} \tag{1}$$

$$\begin{aligned} f(\mathbf{z}_{k+1}, x_{k+1} \,|\mathbf{u}_k) &= \sum_{x_k \in \mathcal{X}} f(\mathbf{z}_{k+1}, x_{k+1} \,|\mathbf{z}_k, x_k, \mathbf{u}_k)\, f(\mathbf{z}_k, x_k \,|\mathbf{u}_k) \\ &= \sum_{x_k \in \mathcal{X}} f(z_{k+1}, x_{k+1} \,|x_k, u_k)\, f(\mathbf{z}_k, x_k \,|\mathbf{u}_{k-1}) \\ &= f(z_{k+1} \,|x_{k+1}) \sum_{x_k \in \mathcal{X}} \Pr(x_{k+1} \,|x_k, u_k)\, f(\mathbf{z}_k, x_k \,|\mathbf{u}_{k-1}) \end{aligned}$$

More on the Belief Update

- The conditional distribution of z_k given x_k is modeled as Gaussian: with mean determined by the ML state of $\mathbf{b}_k$ and variance $\left(T_k^{(s)}\right)^2$
- If the control is **sync**, then no measurement z_{k+1} is available; then, update the belief by applying the Markov model transitions probabilities

$$\mathbf{b}_{k+1} = \mathbf{P} \cdot \mathbf{b}_k$$

Cost Function

- Distance between asset's true location and the ML estimate from the belief state

$$c_k = |L(x_k) - L(\widehat{x}_k)|$$

where

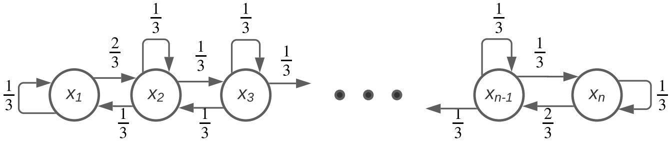
  - x_k is true state
  - $\widehat{x}_k = \arg\max_{x \in \mathcal{X}} b_k(x)$

Movement Models

- Evaluate performance using simple location-only, one-dimensional Markov chains:

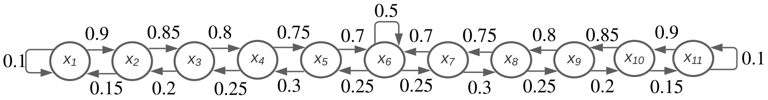
Movement Models

- Evaluate performance using simple location-only, one-dimensional Markov chains:
- Chain 1: uniform probability of staying or moving to adjacent states:

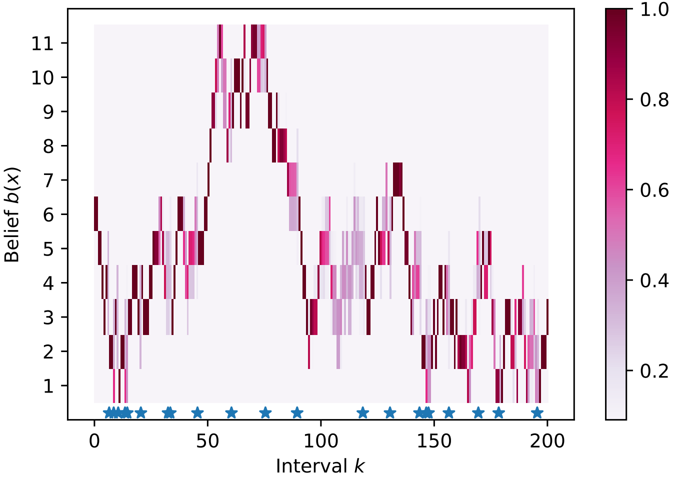


Movement Model 2

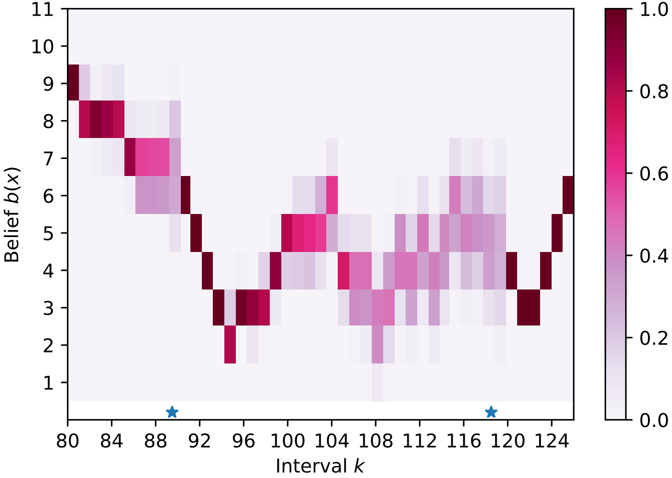
- Chain 2: model an asset that primarily loiters near middle of region, rarely transitions to the outer edges



Belief State Evolution



Belief State Evolution 2



Belief State Compression

- Belief state is a sufficient statistic for deciding the control u_k at stage k

Belief State Compression

- Belief state is a sufficient statistic for deciding the control u_k at stage k
- However: state space has $|\mathcal{X}|$ continuous dimensions
- **Observation:** Beliefs generally concentrated around one state and spread out away from that state
- Quantize beliefs into triple of **discrete** values $\underline{x}_k = [T_k^{(s)}, \hat{x}_k, \sigma_{k,x}^2]$:

Belief State Compression

- Belief state is a sufficient statistic for deciding the control u_k at stage k
- However: state space has $|\mathcal{X}|$ continuous dimensions
- **Observation:** Beliefs generally concentrated around one state and spread out away from that state
- Quantize beliefs into triple of **discrete** values $\underline{x}_k = [T_k^{(s)}, \hat{x}_k, \sigma_{k,x}^2]$:
  - $T_k^{(s)}$: is the number of time since last **sync**
  - $\hat{x}_k$: ML estimate for movement state
  - $\sigma_{k,x}^2$: Quantized variance of movement state

Belief State Compression 2

- Whereas spreading of beliefs is an implicit factor in original belief state, it becomes an explicit component of the compressed state through the variance measure
- Called: *Triple Q-Learning* (TQ-Learning)

TQ-Learning Update

- Use tabular Q-learning with usual update rule:

$$Q(\underline{x}, u) = Q(\underline{x}, u) + \alpha \left[c + \gamma \min_{u'} Q(g(\underline{x}, u), u') - Q(\underline{x}, u) \right]$$

TQ-Learning Update

- Use tabular Q-learning with usual update rule:

$$Q(\underline{x}, u) = Q(\underline{x}, u) + \alpha \left[c + \gamma \min_{u'} Q(g(\underline{x}, u), u') - Q(\underline{x}, u) \right]$$

- Here, c is the cost of performing u from whatever true state the asset actually is in, g is a generic state update function, and u' is the control that minimizes the cost in the next interval

TQ-Learning Update

- Use tabular Q-learning with usual update rule:

$$Q(\underline{x}, u) = Q(\underline{x}, u) + \alpha \left[c + \gamma \min_{u'} Q(g(\underline{x}, u), u') - Q(\underline{x}, u) \right]$$

- Here, c is the cost of performing u from whatever true state the asset actually is in, g is a generic state update function, and u' is the control that minimizes the cost in the next interval
- The other constants affect how learning progresses:

TQ-Learning Update

- Use tabular Q-learning with usual update rule:

$$Q(\underline{x}, u) = Q(\underline{x}, u) + \alpha \left[c + \gamma \min_{u'} Q(g(\underline{x}, u), u') - Q(\underline{x}, u) \right]$$

- Here, c is the cost of performing u from whatever true state the asset actually is in, g is a generic state update function, and u' is the control that minimizes the cost in the next interval
- The other constants affect how learning progresses:
  - α: learning rate

TQ-Learning Update

- Use tabular Q-learning with usual update rule:

$$Q(\underline{x}, u) = Q(\underline{x}, u) + \alpha \left[c + \gamma \min_{u'} Q(g(\underline{x}, u), u') - Q(\underline{x}, u) \right]$$

- Here, c is the cost of performing u from whatever true state the asset actually is in, g is a generic state update function, and u' is the control that minimizes the cost in the next interval
- The other constants affect how learning progresses:
  - α: learning rate
  - γ: discount factor

Stochastic Policies and Model-Free approaches

- Stochastic policies are also optimized over and compared against TQ-learning

Stochastic Policies and Model-Free approaches

- Stochastic policies are also optimized over and compared against TQ-learning
- Fixed-rate stochastic (FRS): controls (u_l, u_s) chosen with probabilities $(1 - \mathit{sync\_rate}, \mathit{sync\_rate})$, respectively

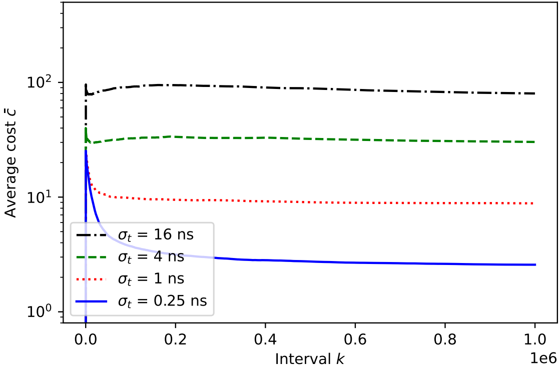
Stochastic Policies and Model-Free approaches

- Stochastic policies are also optimized over and compared against TQ-learning
- Fixed-rate stochastic (FRS): controls (u_l, u_s) chosen with probabilities $(1 - sync\_rate, sync\_rate)$, respectively
  - experimentally found best sync rate to minimize average cost

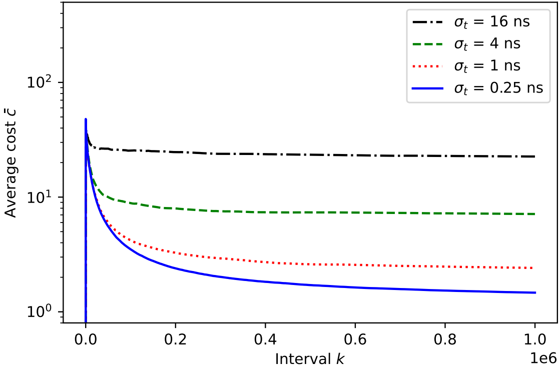
Stochastic Policies and Model-Free approaches

- Stochastic policies are also optimized over and compared against TQ-learning
- Fixed-rate stochastic (FRS): controls (u_l, u_s) chosen with probabilities $(1 - sync\_rate, sync\_rate)$, respectively
  - experimentally found best sync rate to minimize average cost
- Model-free (MF) localization: based on raw localization results from triangulation

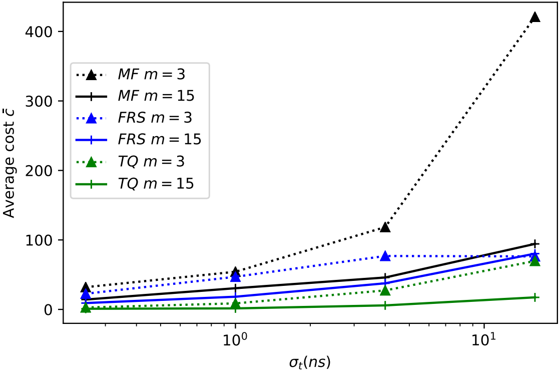
Training Curves: P_1, $m = 3$



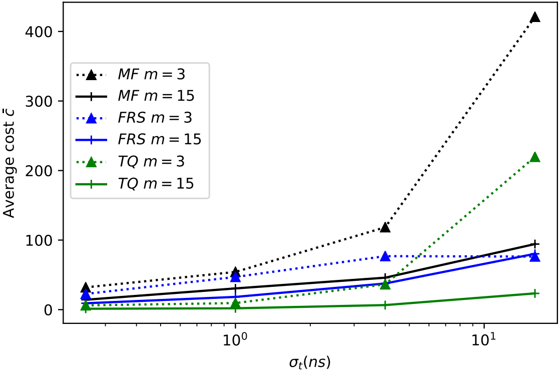
Training Curves, P_2, $m = 15$



Testing Results: Model P1



Testing Results: Model P2



Conclusion

- Formulated problem of optimizing synchronization times for system of distributed sensors tracking an asset as a POMDP

Conclusion

- Formulated problem of optimizing synchronization times for system of distributed sensors tracking an asset as a POMDP
- Applied state-space compression to form low-dimensionality, discrete state space appropriate for tabular Q-learning

Conclusion

- Formulated problem of optimizing synchronization times for system of distributed sensors tracking an asset as a POMDP
- Applied state-space compression to form low-dimensionality, discrete state space appropriate for tabular Q-learning
- Results show Q-learning is able to significantly outperform pure localization or stochastic updates

Conclusion

- Formulated problem of optimizing synchronization times for system of distributed sensors tracking an asset as a POMDP
- Applied state-space compression to form low-dimensionality, discrete state space appropriate for tabular Q-learning
- Results show Q-learning is able to significantly outperform pure localization or stochastic updates
  - can identify when synchronization is needed based on spread of beliefs (as measured through variance)

Conclusion

- Formulated problem of optimizing synchronization times for system of distributed sensors tracking an asset as a POMDP
- Applied state-space compression to form low-dimensionality, discrete state space appropriate for tabular Q-learning
- Results show Q-learning is able to significantly outperform pure localization or stochastic updates
  - can identify when synchronization is needed based on spread of beliefs (as measured through variance)
- Very early work: good candidate for deep Q-learning, want to consider RADAR problem, moving sensors, ...