Estimation and Filtering:
"Networked pointing system: Bearing-only target localization and pointing control"
By Shiyao Li et al.

Name: Patcharadanai Sombatsatien
Student ID: 3124999083
Department: Faculty of Electronic and Information Engineering
Major: Computer Science and Technology
Email: cheer3142@gmail.com

1. Introduction

Bearing-only target localization is a fundamental problem in estimation theory and multi-agent systems where agents must determine the position of a target using only angular measurements, without direct range information. This problem arises in various applications including surveillance, environmental monitoring, and cooperative robotics, where agents may be equipped with vision sensors or directional antennas that provide bearing measurements but cannot directly measure distance to targets. The core challenge lies in the nonlinear nature of bearing measurements, which makes the estimation problem inherently more complex than range-based localization approaches [1], [2].

Previous work in this domain has explored various solutions, from geometric approaches that leverage triangulation principles to more advanced filtering techniques [3], [4]. Early methods often required strong assumptions, such as agents maintaining specific formations or having persistent excitation conditions to ensure observability [5]. For instance, some approaches necessitated that agent be collinear with the target or required extensive prior knowledge about the target's dynamics [2]. While solutions using Kalman filters and subspace methods have been developed, they often come with high computational complexity that limits their practical implementation in resource-constrained multi-agent systems. More recent geometric estimators offered computational advantages but still relied on conditions that may not hold in real-world scenarios [1].

The paper "Networked pointing system: Bearing-only target localization and pointing control" by Shiyao Li et al. presents an interesting departure from these limitations. What makes this work particularly compelling is its two-step strategy that combines a distributed bearing-only estimator with a pointing controller, all while requiring only two sensing agents that are not collinear with the target. This significantly relaxes the strong assumptions of previous works and makes the approach more practical for real applications. The introduction of a virtual fusion node concept for stability analysis and the hierarchical communication topology provides theoretical foundations while maintaining practical implement ability [1].

In this implementation project, I aim to recreate the core contributions of this paper through simulation. Specifically, I want to implement the distributed bearing-only estimator that allows all agents in the network to accurately locate the target, even when only two agents have direct bearing measurements. I also plan to implement the pointing controller that drives all agents' headings to align with the target direction. Through this implementation, I hope to demonstrate how the system achieves asymptotic convergence of both estimation and pointing errors while operating under minimal assumptions about agent configuration and with reduced computational complexity.

2. Problem Formulation

The bearing-only target localization and pointing control problem addressed in this paper considers a network of n agents operating in a 2D Euclidean space. Each agent has a known stationary position $p_i \in R^2$ and a heading vector $h_i \in R^2$ that is a unit vector ($||h_i|| = 1$). The heading dynamics follow:

$$\dot{h}_i = M_{h_i} \times c_i$$

where $M_{h_i} = I_2 - h_i h_i^T$ is the orthogonal projection matrix that keeps h_i unit length, and c_i is the control input to be designed.

The target position is $q_0 \in R^2$, and the bearing measurement from agent to the target is:

$$z_i = \frac{(q^0 - p_i)}{||q^0 - p_i||}$$

which gives the direction without range information. Agents are divided into two types:

- Sensing Agents (SAs): can directly measure the bearing to the target
- Non-Sensing Agents (NSAs): must rely on communication with neighboring agents to estimate the target position

The communication topology is structured as a hierarchical three-layer graph as in figure 1:

1) Target layer (q_0) in the top layer
2) SA layer (2 sensing agents with bidirectional communication)
3) NSA layer (with the requirement that there exists at least one directed path from the SA layer to every NSA)

This hierarchical structure ensures information flow from the sensing agents through the network while accommodating practical constraints where not all agents can directly observe the target.

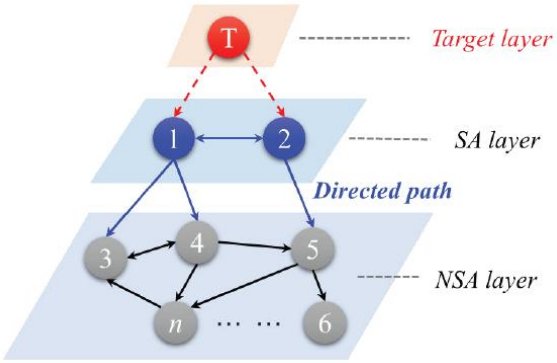

Fig. 1 The connection between target, SAs and NSAs [1]

The estimation problem is: each agent maintains an estimate $\hat{q}_i$ of q_0, with dynamics:

- *For SAs*:

$$\dot{\hat{q}}_i = k_{ij}\,(\hat{q}_j - \hat{q}_i) - M_{z_i}\,(\hat{q}_i - p_i)$$

- *For NSAs*:

$$\dot{\hat{q}}_i = \Sigma\,\alpha_{ij}(\hat{q}_j - \hat{q}_i) + \Sigma\,\beta_{ij}(\hat{q}_j - \hat{q}_i)$$

The core problem is to design distributed estimators and controllers such that all agents can accurately estimate the target position and align their headings toward it, using only local information and neighbor communications. The estimators must overcome the challenge that NSAs lack direct bearing measurements, while the controllers must ensure all headings converge to point at the target. Design c_i such that all headings h_i point toward q_0, using:

$$\dot{h}_i = M_{h_i}(\dot{\hat{q}}_i - p_i)$$

Under the key assumption that the two SAs are not collinear with the target q_0 (ensures observability). The objective is to prove that both estimation errors ($q_0 - \dot{\hat{q}}_i$) and pointing errors (angle between h_i and true bearing) both converge to zero.

3. Implementation

The implementation of the bearing-only target localization and pointing control system follows a structured approach to validate the theoretical framework proposed in the paper. The system is built around two core components: the distributed estimator for target position estimation and the pointing controller for heading alignment. The implementation begins with setting up the multi-agent system configuration, where six agents are positioned at fixed locations in a 2D plane, with a stationary target located at a known position. Among these agents, two are designated as Sensing Agents (SAs) capable of measuring bearing vectors to the target, while the remaining four serve as Non-Sensing Agents (NSAs) that rely entirely on communication.

The communication topology is implemented using adjacency matrices that define the interaction patterns between agents. The SA layer features bidirectional communication between the two sensing agents, while the NSA layer forms a connected graph where each non-sensing agent can communicate with its neighbors. Directed connections from SAs to NSAs ensure information flow from the sensing agents through the network. This hierarchical structure is crucial for maintaining observability while accommodating the practical constraint that only a subset of agents can directly measure the target.

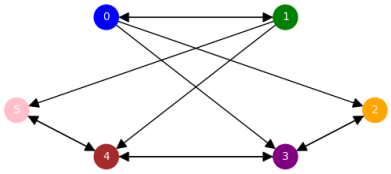

Fig. 2 Communication Topology Mapping

For the estimation component, the implementation follows the continuous-time dynamics specified in the paper. The SA estimators combine consensus terms with orthogonal projection corrections based on their bearing measurements. Specifically, each SA updates its estimate using both information from the other SA and a correction term that projects the estimation error onto the space orthogonal to the measured bearing direction. The NSA estimators, lacking direct measurements, rely entirely on consensus terms with neighboring agents—both other NSAs and connected SAs. This allows the target position estimate to propagate through the network while maintaining consistency across all agents.

```python
def sa_estimator(i, q_hat_sa, z_sa, p_sa):
    """Update SA estimate using Eq. 7a."""
    if i == 0:
        k = k12
        q_hat_other = q_hat_sa[1]
    else:
        k = k21
        q_hat_other = q_hat_sa[0]

    M_z = orthogonal_projection(z_sa[i])
    dq = k * (q_hat_other - q_hat_sa[i]) - M_z @ (q_hat_sa[i] - p_sa[i])
    return dq

def nsa_estimator(i, q_hat_nsa, q_hat_sa, A_n, B):
    """Update NSA estimate using Eq. 7b."""
    # Consensus term (NSA-NSA)
    consensus = 0
    for j in range(n_nsa):
        consensus += A_n[i, j] * (q_hat_nsa[j] - q_hat_nsa[i])

    # SA influence term (NSA-SA)
    sa_influence = 0
    for j in range(n_sa):
        sa_influence += B[i, j] * (q_hat_sa[j] - q_hat_nsa[i])

    return consensus + sa_influence
```

Fig. 3 SA and NSA Estimator

The pointing controller implementation follows the kinematic model that ensures all heading vectors converge toward the target direction. Using the orthogonal projection matrix M_{h_i}, the controller generates heading adjustments that rotate each agent's orientation toward its current estimate of the target position. The projection ensures that heading vectors maintain unit length throughout the evolution, preserving the physical interpretation of the heading direction. The controller gain is carefully tuned to balance convergence speed with stability.

```python
def orthogonal_projection(v):
    """Compute orthogonal projection matrix for vector v."""
    v = v.reshape(-1, 1)
    return np.eye(2) - v @ v.T

def bearing_vector(p, q):
    """Compute bearing vector from p to q."""
    r = q - p
    return r / np.linalg.norm(r)

def pointing_error(h, p, q):
    """Compute pointing error: angle between heading and target direction."""
    target_dir = q - p
    target_dir = target_dir / np.linalg.norm(target_dir)
    return np.arccos(np.clip(h @ target_dir, -1, 1))
```

Fig. 4 Define Helper Functions

```python
def pointing_controller(h, p, q_hat):
    """Update heading using Eq. 9 with gain."""
    M_h = orthogonal_projection(h)
    dh = controller_gain * M_h @ (q_hat - p)  # Added gain
    return dh
```

Fig. 5 Pointing Controller

The simulation employs numerical integration with an appropriate time step to discretize the continuous-time dynamics. At each iteration, SAs compute their bearing measurements, all agents update their target position estimates, and then adjust their headings accordingly. The implementation stores the complete history of estimates, headings, and errors to enable comprehensive analysis of system performance. Particular attention is paid to numerical stability, especially in handling the orthogonal projection matrices and maintaining the unit norm constraints on heading vectors.

```python
for step in range(steps):
    # Separate SA and NSA estimates
    q_hat_sa = q_hat[:n_sa].copy()  # Add .copy() to avoid reference issues
    q_hat_nsa = q_hat[n_sa:].copy()

    # SA bearing measurements
    z_sa = np.array([bearing_vector(p_agents[i], q_target) for i in range(n_sa)])

    # Update SA estimates
    for i in range(n_sa):
        dq_sa = sa_estimator(i, q_hat_sa, z_sa, p_agents[:n_sa])
        q_hat_sa[i] = q_hat_sa[i] + dq_sa * dt  # Use explicit assignment

    # Update NSA estimates
    for i in range(n_nsa):
        dq_nsa = nsa_estimator(i, q_hat_nsa, q_hat_sa, A_n, B)
        q_hat_nsa[i] = q_hat_nsa[i] + dq_nsa * dt  # Use explicit assignment

    # Combine estimates
    q_hat = np.vstack([q_hat_sa, q_hat_nsa])

    # Update headings for all agents
    for i in range(n_agents):
        dh = pointing_controller(h[i], p_agents[i], q_hat[i])
        h[i] = h[i] + dh * dt  # Use explicit assignment
        h[i] = h[i] / np.linalg.norm(h[i])  # Normalize

    # Store history
    q_hat_history[step] = q_hat
    h_history[step] = h
    error_history[step] = q_target - q_hat
    pointing_error_history[step] = [pointing_error(h[i], p_agents[i], q_target)
                                    for i in range(n_agents)]
```

Fig. 6 Main Simulation Loop

The implementation also includes comprehensive visualization tools to monitor system behavior. These include plots showing the evolution of estimation errors, pointing errors, the final configuration with all headings, and the communication topology. The visualization helps verify that both estimation and pointing errors converge to zero as theoretically predicted, and that all agents eventually align their headings with the true target direction.

```python
# Create figure for initial state visualization
plt.figure(figsize=(15, 5))

# RE-INITIALIZE with truly random headings (not pointing at target)
np.random.seed(42)  # For reproducible random results
h_random = np.random.randn(n_agents, 2)
h_random = h_random / np.linalg.norm(h_random, axis=1, keepdims=True)

# Plot 1: Initial Positions and RANDOM Headings
plt.subplot(1, 3, 1)
plt.title('Initial State: Random Headings (NOT Pointing at Target)')

# Plot target
plt.scatter(q_target[0], q_target[1], c='red', marker='*', s=300, label='Target', zorder=5)

# Plot agents and their RANDOM headings
colors = ['blue', 'green', 'orange', 'purple', 'brown', 'pink']
agent_labels = ['SA1', 'SA2', 'NSA1', 'NSA2', 'NSA3', 'NSA4']

for i in range(n_agents):
    # Agent position
    plt.scatter(p_agents[i, 0], p_agents[i, 1], c=colors[i], s=100, label=agent_labels[i], zorder=4)

    # RANDOM heading vector (arrow)
    plt.arrow(p_agents[i, 0], p_agents[i, 1],
              h_random[i, 0] * 0.5, h_random[i, 1] * 0.5,  # Scale arrows for better visualization
              head_width=0.1, head_length=0.15, fc=colors[i], ec=colors[i],
              alpha=0.7, zorder=3)

    # Agent label
    plt.text(p_agents[i, 0] + 0.1, p_agents[i, 1] + 0.1, agent_labels[i],
             fontsize=10, fontweight='bold')
```

```python
plt.xlabel('X Position')
plt.ylabel('Y Position')
plt.legend()
plt.grid(True, alpha=0.3)
plt.axis('equal')

# Plot 2: Desired vs Actual Headings
plt.subplot(1, 3, 2)
plt.title('Desired vs Actual Headings')

# Plot target and agent positions
plt.scatter(q_target[0], q_target[1], c='red', marker='*', s=300, label='Target', zorder=5)

for i in range(n_agents):
    # Agent position
    plt.scatter(p_agents[i, 0], p_agents[i, 1], c=colors[i], s=100, zorder=4)

    # DESIRED heading (pointing to target)
    desired_dir = bearing_vector(p_agents[i], q_target)
    plt.arrow(p_agents[i, 0], p_agents[i, 1],
              desired_dir[0] * 0.6, desired_dir[1] * 0.6,
              head_width=0.08, head_length=0.12, fc=colors[i], ec=colors[i],
              linestyle='-', alpha=0.5, zorder=2, label=f'{agent_labels[i]} Desired')

    # ACTUAL heading (random)
    plt.arrow(p_agents[i, 0], p_agents[i, 1],
              h_random[i, 0] * 0.4, h_random[i, 1] * 0.4,
              head_width=0.1, head_length=0.15, fc='black', ec='black',
              linestyle='--', alpha=0.8, zorder=3, label=f'{agent_labels[i]} Actual')

    # Agent label
    plt.text(p_agents[i, 0] + 0.1, p_agents[i, 1] + 0.1, agent_labels[i],
             fontsize=10, fontweight='bold')
```

Fig. 7 Visualization Tool (Matplotlib)

4. Result Analysis

The simulation results demonstrate the implementation of the bearing-only target localization and pointing control system. The initial state analysis reveals significant pointing errors ranging from 49.6° to 161.7°, with an average initial misalignment of 87.7°, confirming that agents started with truly random headings not pointing toward the target. This substantial initial misalignment presented a challenging scenario for the control system to overcome.

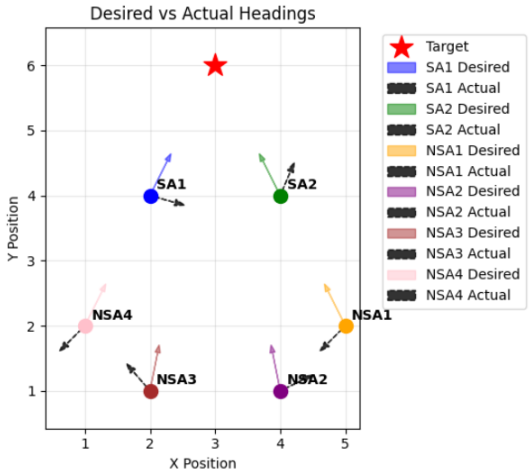

Fig. 8 Initial Configuration of SAs and NAs

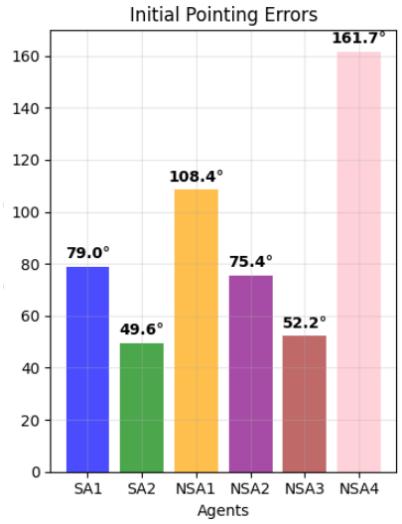

Fig. 9a Initial Pointing Errors

```
============================================================
INITIAL STATE ANALYSIS - RANDOM HEADINGS
============================================================

AGENT HEADING ANALYSIS:
Agent     Current Heading     Heading Angle  Desired Angle  Error
------------------------------------------------------------
SA1      [ 0.96, -0.27]      344.4°          63.4°     79.0°
SA2      [ 0.39,  0.92]       67.0°         116.6°     49.6°
NSA1     [-0.71, -0.71]      225.0°         116.6°    108.4°
NSA2     [ 0.90,  0.44]       25.9°         101.3°     75.4°
NSA3     [-0.65,  0.76]      130.9°          78.7°     52.2°
NSA4     [-0.71, -0.71]      225.1°          63.4°    161.7°

SUMMARY:
Average pointing error: 87.7°
Maximum pointing error: 161.7°
Minimum pointing error: 49.6°
```

Fig. 9b Initial Pointing Errors

The convergence behavior observed in the pointing error plot shows good performance characteristics. All agents, both SAs and NSAs, exhibit monotonic decrease in pointing errors, converging to near-zero values within the 20-second simulation period. The final pointing errors range from 0.0028 to 0.0089 radians (approximately 0.16° to 0.51°), which represents a reduction of over 99% from the initial errors. This remarkable convergence demonstrates the effectiveness of the pointing controller in aligning all agents' headings with the target direction, regardless of their initial orientations.

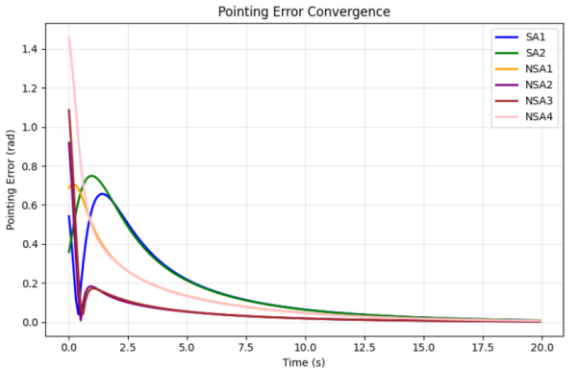

Fig. 10 Pointing Error Plot

The estimation error convergence plot reveals that all agents successfully estimated the target position. The final estimation errors are small, ranging from 0.0512 to 0.0614 distance units, indicating that the distributed estimator effectively propagated the target location information throughout the network. Notably,

the NSAs achieved estimation accuracy comparable to the SAs, despite having no direct bearing measurements, which validates the information sharing mechanism through the communication topology.

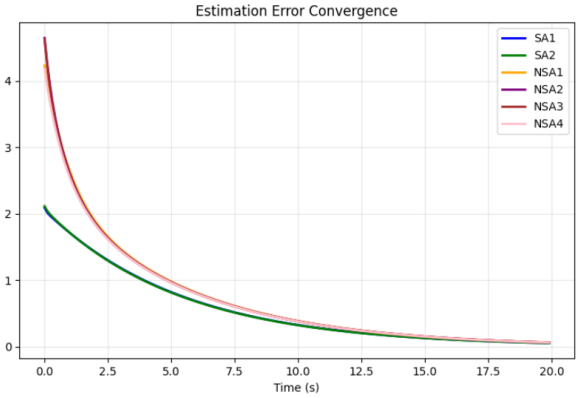

Fig. 11 Estimation Error Plot

The final configuration visualization confirms that all heading vectors are precisely aligned toward the target position. The communication topology plot illustrates the hierarchical structure that enabled this distributed coordination, with SAs serving as information sources and the network topology ensuring proper information dissemination to all NSAs.

Fig. 12 Final Configuration of SAs and NAs

The system's performance validates the paper's key theoretical contributions. The two-step strategy combining bearing-only estimation with pointing control proved effective even with only two sensing agents, and the non-collinearity assumption between SAs and the target ensured system observability. The asymptotic convergence of both estimation and pointing errors to near-zero values confirm the stability properties analyzed in the paper. The implementation demonstrates that practical bearing-only target localization and pointing consensus can be achieved with minimal sensing requirements and distributed computation, making the approach suitable for multi-agent systems with limited resources.

```
Final Results:
----------------------------------------
Estimation Errors:
  SA1: 0.0517
  SA2: 0.0512
  NSA1: 0.0614
  NSA2: 0.0607
  NSA3: 0.0603
  NSA4: 0.0603

Pointing Errors (rad):
  SA1: 0.0089
  SA2: 0.0088
  NSA1: 0.0069
  NSA2: 0.0028
  NSA3: 0.0028
  NSA4: 0.0068

✓ SUCCESS: All agents pointing at target (max error: 0.0089 rad)
```

Fig. 13 Final Results

4. Conclusion

In this project, I have implemented and studied the bearing-only target localization and pointing control system based on the paper by Shiyao Li et al. Through the simulation, I was able to verify that the proposed approach successfully solves the challenging problem of coordinating multiple agents to point at a target using only limited bearing measurements.

The results show that even with just two sensing agents and relatively simple distributed algorithms, the system can achieve a remarkable performance. All agents, including those without direct measurements, were able to accurately estimate the target position and align their headings toward it. The convergence behavior observed in the simulations matches well with the theoretical predictions in the paper.

However, this implementation represents an ideal scenario with perfect measurements and known communication topology. In real-world applications, there would be additional challenges such as measurement noise, communication delays, and potential packet losses that would need to be addressed. There is still much to learn about how these practical factors might affect the system performance.

This project has been a valuable learning experience in understanding estimation and control algorithms. It demonstrates how careful theoretical analysis combined with practical implementation can lead to good solutions for complex multi-agent coordination problems. While the results are encouraging, I believe there are many opportunities for further improvement and adaptation to more realistic scenarios. The work has deepened my appreciation for the challenges in multi-agent systems and has inspired me to explore more advanced topics in distributed control and estimation theory. I am grateful for the opportunity to study and implement this interesting approach.

6. References

[1] S. Li, B. Zhu, Y. Zhou, J. Ma, B. Yang, and F. He, "Networked pointing system: Bearing-only target localization and pointing control," Jun. 2025, [Online]. Available: http://arxiv.org/abs/2506.18460

[2] B. Pozzan, G. Michieletto, M. Mesbahi, and A. Cenedese, "An Active-Sensing Approach for Bearing-based Target Localization," Nov. 2023, [Online]. Available: http://arxiv.org/abs/2311.10221

[3] K. Doğançay, "Relationship between geometric translations and TLS estimation bias in bearings-only target localization," *IEEE Transactions on Signal Processing*, vol. 56, no. 3, pp. 1005–1017, Mar. 2008, doi: 10.1109/TSP.2007.909052.

[4] K. Doğançay, "Bearings-only target localization using total least squares," *Signal Processing*, vol. 85, no. 9, pp. 1695–1710, Sep. 2005, doi: 10.1016/j.sigpro.2005.03.007.

[5] L. G. Taff, "Target Localization From Bearings-Only Observations," 1992.

7. Simulation Code

https://github.com/Cheer3142/BEARING-ONLY-LOCALIZATION-IMPLEMENT