Autonomous Adaptable Simple Object Transport via Caging with a Homogeneous Multi-Agent Robot Swarm

Daniel Sam, Nicolas Cuneo

November 2025

Contributions of each student are indicated by the star of the corresponding color (⋆⋆).

This report was written with assistance from ChatGPT 5.1 in the following areas: LaTeXformatting, writing editing, simulation code drafting + editing.

I. Abstract ⋆

This paper presents a general algorithm for autonomously and adaptively transporting any object in a 2-dimensional workspace by a homogeneous multi-agent robot swarm. This work contributes to the broader goal of developing "smart manufacturing" systems by addressing the basic task of general object transport and manipulation. This approach involves robots implementing a caging strategy around the object at equidistant points along its perimeter. The object's shape is expressed mathematically through a Signed Distance Function (SDF), which comes from a set of several circular SDFs. In this way, agents determine their placement with respect to the object. First, using the marching squares algorithm on a modified SDF yields a closed contour that is used to determine the agent's initial caging locations. These points are resampled to ensure that they are equidistant around the object's perimeter. Object transport involves a continuous, time-discretised coordinate transform applied to the caging points. This is carried out by linear interpolation for both position and rotation; hence, the swarm will move collectively while retaining the shape of the object from an initial pose to a final goal pose. The result yields a baseline for more generalised object transport problems-involving 3-dimensions or heterogeneous agents; this can be integrated into larger systems for complex object construction.

Introduction ⋆

Assembly lines were invented in 1901 in order to manufacture the first Ford model T. In the last 124 years technological advancements in dozens of fields, and the invention of entirely new realms of technology, has turned the assembly line into a modern manufacturing megaforce, and one of the most important technologies in our society. But even in assembly lines, humans do 71% of tasks. Robot control algorithms are not generalizable or intelligent, they

are simply efficient at repetitive tasks and brute force. In the last 10 years, advances in control theory and machine learning have created new horizons for robotics controls, expanding the possibilities for smart manufacturing and creating the potential of cheaper, higher volume, quicker manufacturing with the same number of humans. A future with intelligent manufacturing lines presents numerous benefits for all people, and more prosperity for all of mankind.

However, this idea of "smart manufacturing" is so expansive that it could not even be considered a single field, it is comprised of new and experimental parts of many advanced fields. Many smaller problems need to be solved to work towards this goal. One such problem is general object transport and manipulation. This very basic task is one of many simple actions that a team of robotic agents would need to be capable of in a general sense for smart manufacturing. The goal of this project is to produce a general algorithm for the transport of any object in 2-dimensions via a swarm of N robots. This is one possible interpretation of the object transport problem, but an important exercise in developing the math necessary to move on to more general object transport (ex 3-dimensions, heterogeneous agents). A sufficiently general and intelligent algorithm for object transport could be applied in a larger algorithm for object construction. Combining these algorithms with machine learning and complex vision systems also creates the potential for even more general object transport, manipulation, and eventually construction.

II. Mathematical Model ⋆

Workspace $W \subset \mathbb{R}^2$, rectangular, of any size, without obstacles.

N robots at positions:

$$q_i = (q_{ix}, q_{iy}) \in W$$

have velocities:

$$\dot{q_i} = (\dot{q_{ix}}, \dot{q_{iy}}) \in [-1\frac{m}{s}, 1\frac{m}{s}]$$

and accelerations:

$$\ddot{q_i} = (\ddot{q_{ix}}, \ddot{q_{iy}}) \in [-1\frac{m}{s^2}, 1\frac{m}{s^2}]$$

Agents begin with knowledge of environment size and shape, their starting location, and the pose and shape of the transport object. All agents also begin with information on the goal pose for the transport object. Agents can apply force $f_i \in (0, 1N)$, in forward direction. Agents cannot apply sideways force, nor can they pull on the transport object.

Agents can communicate with all other agents within a radius $R_{com} > 0$ around them. At any discrete time k, the robot positions are $q_i(k) \in \mathbb{R}^2$, then the communication graph is:

$$\mathcal{G}(k) = (\mathcal{V}, \mathcal{E}(k))$$

with vertices:

$$\mathcal{V} = 1, 2, ..., N$$

and edges:

$$(i, j) \in \mathcal{E}(k)$$

Thus if $||q_i(k) - q_j(k)|| \leq R_{com}$ then robots i and j can communicate.

The foreman will have a complete view of the workspace, and update the pose of the transport object at discrete time intervals, communicating the updated pose with all agents. The foreman will not track the agents or communicate with the agents about other agent locations.

Let the transport object be any object that fits within the workspace, of a known material. The object's shape will be represented mathematically with a Signed Distance Function (SDF) that will be constructed out of the union of many circular SDFs that follow the perimeter of the object, similar in spirit to implicit–surface swarm control approaches such as [1]. In this paper we will assume the SDF has already been computed.

If circle i is located at $c_i = (c_{ix}, c_{iy}) \in W$ then it's SDF is $\alpha_i(x) = ||x - c_i|| - r$ where r is it's radius about c_i and $x \in \mathbb{R}^2$. n circles can be combined with the smooth minimization function:

$$\phi(x) = smoothmin(\alpha_1, \alpha_2, ...\alpha_n) = -\frac{1}{k} \log \Big(\sum_{i=1}^{n} e^{-k\alpha_i} \Big) \tag{1}$$

which is a standard smooth approximation of the minimum often used in implicit representations of complex shapes [1].

This SDF is generalizable for n circles of the same or varying radii, and will produce a function with the following properties:

$$\begin{cases} \phi(x) > 0, & \text{Outside object} \\ \phi(x) = 0, & \text{Edge of object} \\ \phi(x) < 0, & \text{Inside object} \end{cases}$$

The actual object itself is $\Phi = \{x \in W \,|\, \phi(x) \leq 0\}$ where $x = [x, y]^T \in \mathbb{R}^2$. Assuming Φ has constant density ρ, the total mass is given by:

$$m = \int_{\Phi} \rho dA = \rho \int_{\Phi} dA$$

Define an indicator of "inside the object" using Heaviside:

$$H(-\phi(x)) = \begin{cases} 1, \ \phi(x) \leq 0 \\ 0, \ \phi(x) > 0 \end{cases}$$

Then m in terms of the full SDF is:

$$m = \int_{W} \rho \, H\big(-\phi(x)\big) \, dA \tag{2}$$

The center of mass of object Φ is:

$$x_{com} = \frac{1}{m} \int_{\Phi} x \rho dA$$

Rewrite it over the whole workspace using Heaviside like (2):

$$x_{com} = \frac{1}{m} \int_{W} x \rho H(-\phi(x)) dA$$

Then substitute in (2) to get:

$$x_{com} = \frac{\int_W x H(-\phi(x)) dA}{\int_W H\big(-\phi(x)\big)} \tag{3}$$

Where $x_{com} \in W \subset \mathbb{R}^2$. The position of agent i in relation to the shape follows:

$$\begin{cases} \phi(q_i) > R_{\text{bot}}, & \text{agent is outside the object,} \\ \phi(q_i) < R_{\text{bot}}, & \text{agent is intersecting the object.} \end{cases}$$

The distance from agent i to the outside of Φ as a scaler is given by $\phi(q_i)$. The unit vector corresponding to this scalar can be found with a normalized gradient:

$$\hat{n} = \frac{\nabla \phi(q_i)}{\|\nabla \phi(q_i)\|} \tag{4}$$

$$\phi(x) = -\frac{1}{k} \log(Z(x))$$

$$\nabla \phi(x) = -\frac{1}{k} \frac{1}{Z(x)} \nabla Z(x)$$

$$\nabla Z(x) = \sum_{i=1}^{n} \nabla e^{-k\alpha_i} = \sum_{i=1}^{n} -k(\nabla \alpha_i e^{-k\alpha_i})$$

$$\nabla \phi(x) = \frac{1}{Z(x)} \sum_{i=1}^{n} e^{-k\alpha_i} \nabla \alpha_i$$

So,

$$\nabla \phi(x) = \sum_{i=1}^{n} w_i(x) \nabla \alpha_i$$

Where the weight function $w_i(x)$ has properties

$$\begin{cases} \sum_{i=1}^{n} w_i(x) = 1, \\ 0 \leq w_i(x) \leq 1 \end{cases}$$

And,

$$\nabla\alpha_i(x) = \frac{x - c_i}{||x - c_i||}$$

Thus plugging all this into (4) we get:

$$\hat{n}(q_i) = \frac{\sum_{i=1}^{N} w_i(q_i)\nabla\alpha_i(q_i)}{||\sum_{i=1}^{N} w_i(q_i)\nabla\alpha_i(q_i)||} \tag{5}$$

And scaled to Φ:

$$\gamma_i = \phi(q_i) * \hat{n}(q_i) \tag{6}$$

This is essentially a weighted average of the normal vectors from each circle that comprises the full shape's SDF, as constructed in (1)–(5). The vector $\hat{p_i}$ begins at agent i and extends to the nearest point on Φ, which provides agent i with full information about its location w.r.t object Φ. Another potentially useful vector is that of agent i's location w.r.t. the COM of object Φ:

$$\gamma_i' = (x_{com} - q_i) \in W \tag{7}$$

These functions will allow each agent to develop it's own map of the world based purely on their own perception and the foreman's information about the location of the object, via (3), (5), and (6)–(7). The robots can communicate their own positions to nearby agents, and as will be proved in the next section the partial communication of the swarm will lead to consensus on the transport object's location and the locations of every agent.

Now the agents can understand the world. But what good is that if they don't know where to go. Initially the hope was to create a generalized physics model that minimized the transport time between the objects initial pose and final pose. The proved more difficult than initially thought, but cooler heads prevailed and a smarter method emerged. Caging. Caging is the process of surrounding an object entirely, which is closely related to formation–shape control around implicit surfaces [1]. This process is much simpler mathematically and should lead to the same result, just not an optimized general solution.

II.I Caging ⋆

First the agents need to cage the object at N evenly spaced points around it's perimeter. An algorithm for this can be developed based on marching squares. The agents will align themselves with the SDF $\psi(x, y) = \phi(x, y) - R_{bot} - \gamma$ where γ is some user chosen spacing function to account for controller and prevent the agents from attempting to intersect the object.

Step 1: Sample $\psi(x, y)$ on a regular grid (x_i, y_i).

Step 2: Run marching squares on ψ_{ij} with isovalue 0
This returns a vector of vertices $[v_0, v_1 ... v_M]$ approximating the closed contour of $\psi(x, y)$.

Step 3: Compute arc length of contour:

$$L = \sum_{j=1}^{N} ||v_{j+1} - v_j||$$

Step 4: Resample N evenly spaced points: For k = 0,1,...,N-1 Target arc length:

$$s_k = \frac{k}{N} L$$

Find j such that:

$$s_j \leq s_k \leq s_{j+1}$$

Where s_j is essentially the arc length up to vertex v_j

$$s_j = \sum_{i=1}^{j} ||v_{i+1} - v_i||$$

Thus we are looking for the j that is immediately before s_k. Then we interpolate:

$$\alpha = \frac{s_k - s_j}{s_{j+1} - s_{j-1}}$$

Then we define:

$$p_k = (1 - \alpha) v_j + \alpha v_{j+1}$$

Thus the vector $P = [p_0, p_1, ..., p_{N-1}]$ is a set of N points evenly spaced around $\phi(x, y)$ where $P \in \mathbb{R}^2$. Each point can be assigned to an agent such that $q_i = p_i$ for all i. These points P will be used in the transport parameterization in Section II.II and in the controller of Section II.III.

II.II Transport ⋆

In order to transport Φ the robots simply need to collectively move in the shape of the object. Section II.I showed how to cage the object by calculating the locations of N evenly spaced points about the perimeter of Φ, P. To achieve transport we simply apply the proper coordinate transform to each point in P such that they retain their shape and move

Φ from it's start position X_i and starting rotation θ_i to it's final position X_f and rotation θ_f. Simply define a rotation matrix:

$$R(\theta) = \begin{pmatrix} cos(\theta_f) & -sin(\theta_f) \\ sin(\theta_f) & cos(\theta_f) \end{pmatrix}$$

And then apply the SE(2) coordinate transform to all points in P:

$$p'_k = X_f + R(\theta_f)(p_k - X_i) \tag{8}$$

However this simply jumps all the points from their initial positions to their final positions. In order to get a continuous path between the two we can discretize time and apply portions of the coordinate transform each step such that the sum total of the many coordinate transforms equals the full transform. We introduce a parameter s and then linearly interpolate between position and rotation:

Position:

$$X(s) = (1 - s)X_i + sX_f \tag{9}$$

Rotation:

$$\theta(s) = (1 - s)\theta_i + s\theta_f \tag{10}$$

Then replace the final goal position X_f and rotation θ_f in equation 8 with 9 and 10 to yield:

$$p^*_k(s) = X(s) + R(\theta(s))(p_k - X_i) \tag{11}$$

Thus equation 11 gives the location of all points at all times between object Φ's initial position and final position.

II.III Potential–Field Based Controller

A decentralized potential-field controller can be developed in order to guide the agents to their caging positions and to maintaining caging formulation during object transport. This potential field will be composed of three terms: (1) attraction to the robot's assigned point on the object boundary, (2) a small repulsion from other agents, and (3) repulsion from the workspace boundary. This type of gradient–based navigation using artificial potentials is closely related to classical navigation function methods [2]. Unlike classic potential field navigation functions there will be no repulsive potential from the object, as the robots must press directly against the object's surface.

(1) Goal Attraction:

$p_i \in \mathbb{R}^2$ denotes the position of robot i, and $p^*_i(s)$ is its assigned caging point on the object boundary parameterized by $s \in [0, 1]$. As derived in the previous section:

$$p^*_i(s) = X(s) + R(\theta(s))(p_i - X_i), \tag{12}$$

We can define the attraction that drives robot i toward p_i^* as:

$$U_{\text{att},i}(p_i) = \frac{1}{2}\|p_i - p_i^*(s)\|^2. \tag{13}$$

Minimizing $U_{\text{att},i}$ corresponds to agent i being on its correct location p_i^* about Φ.

(2) Inter–Robot Repulsion:

To prevent collisions while the agents travel from their initial positions to their caging positions, and even throughout object transport, a short range repulsion between nearby agents is required. An inverse-distance repulsion potential, as used in [3], can be defined:

$$U_{ij}(p_i, p_j) = \frac{k_r}{\|p_i - p_j\|^2}, \qquad k_r > 0. \tag{14}$$

This term acts only when neighbors approach too closely.

Thus the full repulsion felt by agent i due to all other agents is:

$$U_{\text{rep},i}(p) = \sum_{j \neq i} U_{ij}(p_i, p_j). \tag{15}$$

(3) Workspace Boundary Repulsion

We assume the boundary of workspace W is known to be ∂W. To prevent robots from colliding with the wall, we can apply a barrier potential based on their distance $d_i = \text{dist}(p_i, \partial W)$ from the wall:

$$U_{\text{wall},i}(p_i) = \frac{k_w}{d_i^2}, \qquad k_w > 0. \tag{16}$$

This potential activates only when a robot approaches the workspace boundary and is zero elsewhere. k_w is a tunable constant to modulate the strength of this repulsion.

Total Potential and Control Law

Thus derived from (13), (15), and (16) the total potential for robot i is:

$$U_i(p) = U_{\text{att},i}(p_i) + U_{\text{rep},i}(p) + U_{\text{wall},i}(p_i). \tag{17}$$

Each robot applies a decentralized gradient–descent controller

$$\dot{p}_i = -\nabla_{p_i} U_i(p), \tag{18}$$

which expands to

$$\dot{p}_i = -(p_i - p_i^*(s)) - \sum_{j \neq i} \nabla_{p_i} U_{ij}(p_i, p_j) - \nabla_{p_i} U_{\text{wall},i}(p_i). \tag{19}$$

This yields three interpretable motion components:

- **Goal Position Attraction:** Pushes agent i toward the correct point on the transported object's boundary according to the mathematical model derived above.

- **Inter-Agent Collision Avoidance:** Repels robot i from neighbors if they approach too closely through equation (14).
- **Workspace safety:** Keeps robot i away from the outer wall using equation (16).

Discrete–Time Implementation

For a simulation, or real world implementation, the continuous controller must be made discrete. This is done using Euler integration with timestep $\Delta t > 0$:

$$p_i(k+1) = p_i(k) + \Delta t\, \dot{p}_i\big(p(k)\big). \tag{20}$$

And there you have it. That is your controller. Because all terms depend only on the local neighbor positions, and the known trajectory, (11), the controller is scalable and fully decentralized.

As a short recap, our object is defined with a shaped distance function. N points at some adequate distance away from the objects boundary are calculated based on this SDF, and then all N agents find their positions about the object. These points are then moved with a parametrized coordinate transform from the transport objects initial position to it's final position, maintaining the caged shape the whole way through. The robots are controlled using a potential field based controller that attracts them to their respective points, and repels them from nearby agents and the boundary.

III. Theoretical Analysis ⋆

A few theoretical analyses can be run on the mathematical model described in Section II. to verify its functionality. The entire system is predicated on proper communication between the agents, which was barely touched on in the previous section. The communication graph network outlined above can be analyzed to determine if the robots will achieve consensus on the objects location and their respective locations under the communication conditions, following standard consensus formulations [4, 5] and flocking-style interaction rules [6].

We consider two consensus phases: one before the robots reach their assigned caging points (when the communication graph is changing), and one after the robots are arranged around the object (when the graph is essentially fixed). We then show why the potential–field controller keeps the formation stable during motion.

At any time s, robots can communicate with neighbors inside R_{com}, forming a time–varying graph $\mathcal{G}(s) = (\mathcal{V}, \mathcal{E}(s))$. Before the agents settle into their caging positions the graph changes, but after caging is formed it remains fixed and ring–shaped.

(1) Pre-Caging Consensus Under Switching Graphs

During exploration, each robot maintains estimates of quantities such as the object pose, nearby agent locations, or expected caging positions. Let $z_i(k)$ be one such estimate held by robot i at discrete time k. Collecting all estimates gives $z(k) \in \mathbb{R}^N$.

Robots update their estimates by averaging with their current neighbors:

$$z(k+1) = W(k)\, z(k), \qquad W(k) = I - \varepsilon L(k), \tag{21}$$

where $L(k)$ is the Laplacian of the communication graph at time k. For sufficiently small ε, $W(k)$ behaves like a weighted average between each robot and its neighbors, as in standard consensus schemes [4, 5].

Although the graph may change from step to step, the key requirement is that robots do not break into permanently disconnected subgroups. As long as the group remains "jointly connected" over time windows, the repeated averaging causes all robots to gradually move toward the same value. Any differences in initial estimates are repeatedly blended through neighbor interactions until a common estimate emerges.

This behavior implies that before the caging formation is created, the team can still synchronize information such as the object's reported location and rough group structure, even though no fixed communication pattern exists.

(2) Consensus During Transport Under a Fixed Ring Graph

Once the robots reach their assigned caging points, their spacing and neighbors remain constant. The communication graph becomes essentially a ring, where each robot communicates with its two nearest neighbors, similar to ring and lattice structures studied in distributed flocking and formation control [6].

With a fixed graph, the consensus update becomes

$$z(k+1) = Wz(k), \qquad W = I - \varepsilon L, \tag{22}$$

where L is now constant. Because the graph is connected, each update step reduces the differences between neighboring robots. Repeated application smooths the entire state vector until all robots settle on the same value.

For fixed graphs, the long–term value is simply the average of the initial values in $z(0)$. This ensures that once the formation is established, the robots maintain a consistent shared estimate of any world–model quantity during transport.

(3) Stability of the Potential–Field Transport Controller

The potential–field controller developed in Section II.III ensures that each robot seeks its assigned caging position while avoiding collisions with other agents and the workspace boundary. The controller is built from three potentials: (1) attraction to the desired point on the moving object boundary, (2) short–range repulsion from neighboring robots, and (3) wall repulsion. This follows the general idea of using artificial potential functions to encode navigation and obstacle avoidance [2], combined with formation–shape maintenance similar to [1, 3].

For robot i, the combined potential is

$$U_i(p) = U_{\text{att},i}(p_i) + U_{\text{rep},i}(p) + U_{\text{wall},i}(p_i).$$

To study how the formation behaves, we consider the total energy

$$V(p) = \sum_{i=1}^{N} U_i(p).$$

This value is low when robots are near their correct caging positions and high when robots are far from where they should be or too close to one another.

Under the gradient–descent control law

$$\dot{p}_i = -\nabla_{p_i} U_i,$$

the value of $V(p)$ always decreases or stays the same. This means the system naturally moves toward configurations where the forces on each robot balance out.

Such balanced configurations include the desired caging formation following the SE(2) trajectory. In this formation, each robot is at its assigned boundary point and repulsive forces are inactive. Because the total energy cannot increase, disturbances or small deviations will be corrected as the robots move back toward low–energy states.

Overall, the potential–field controller keeps the caging formation organized during transport. Robots follow their assigned points on the object boundary, maintain spacing, and avoid collisions, all while the object moves along its planned path.

IV. Simulation ⋆⋆

To evaluate the proposed caging and transport strategy, a full multi–robot simulation was built in MATLAB. Each robot is modeled as a kinematic point that moves according to a potential–field controller. Before caging occurs, the robots use an object–avoidance potential to prevent them from cutting through the object as they spread out. Once the agents reach a shell around the object, the controller switches to a caging potential that equalizes their spacing. Finally, after the cage forms, the swarm tracks the object's prescribed SE(2) motion while maintaining formation.

(1) Formation of the Cage

The robots start from random initial conditions and move toward the object boundary without penetrating it. The avoidance field successfully keeps all agents outside the interior while allowing them to settle into a ring-shaped distribution. Figure 1 shows the early stage of this process.

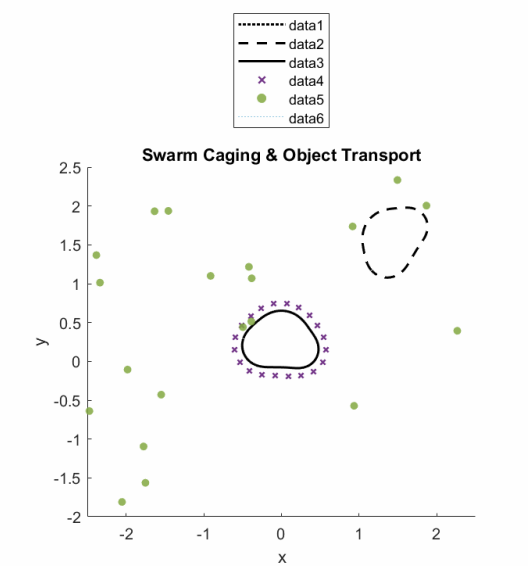

Figure 1: Initial configuration of the swarm and object. Robots begin far from the object and use the avoidance potential to move toward the boundary without crossing it.

As the agents approach the boundary, the controller transitions into the caging potential. This causes the robots to spread evenly around the object and stabilize into a circular formation. The reduced marker sizes and expanded workspace in the visualization make this behavior easier to see. A representative frame is shown in Fig. 2.

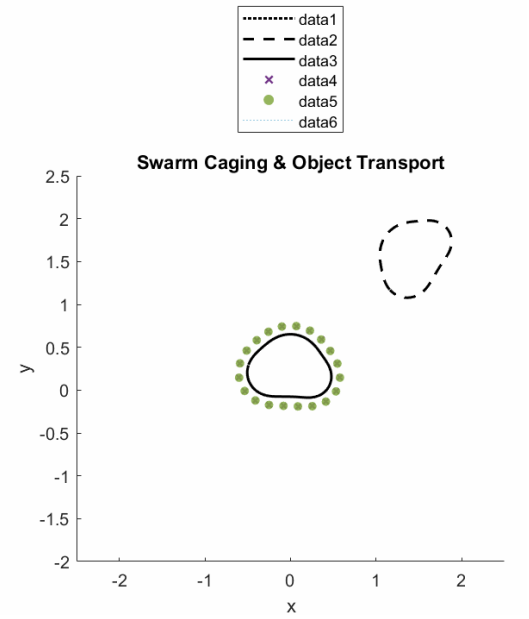

Figure 2: Robots converging from random starting positions into a stable, evenly spaced caging formation around the object.

(2) Consensus Graph Behavior

Throughout the simulation, robots also run a consensus update with their neighbors. Earlier versions of the simulator showed unstable graph rendering, but after fixing the graph indexing, the consensus weights now evolve smoothly without frame-to-frame jumping. A snapshot of this behavior is shown in Fig. 3.

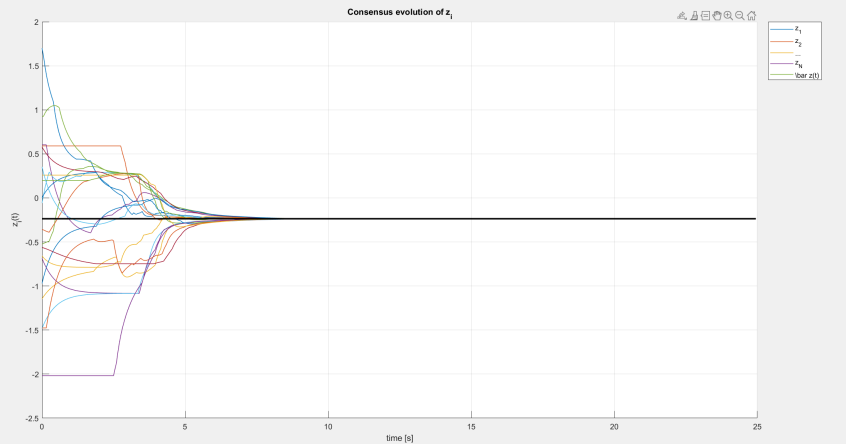

Figure 3: Consensus weight matrix visualization. The updated plotting method eliminates the jitter seen in earlier simulations.

(3) Transport of the Object

Once the cage is established, the object begins following a planned translational and rotational trajectory. The robots do not exert physical forces in this simulation; instead, they track the object's motion while maintaining their relative spacing. This gives a clean demonstration of formation maintenance during transport.

Figure 4 shows an intermediate frame of the object translating across the workspace while the robots keep the cage intact.

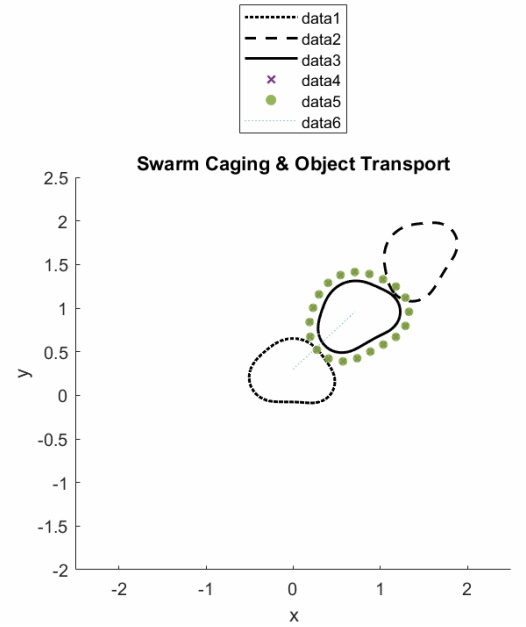

Figure 4: Caged transport in progress. The swarm translates with the object while preserving an even formation.

A final snapshot of the completed motion is given in Fig. 5. The robots maintain their spacing and orientation relative to the object throughout the full path.

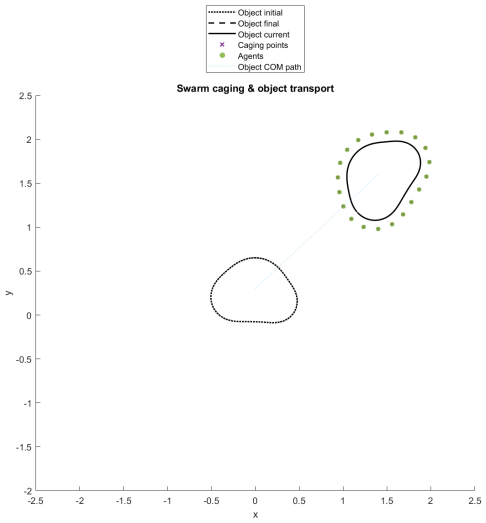

Figure 5: Final position of the object and the surrounding cage after the full SE(2) motion.

(4) Quantitative Results

To better understand the behavior of the swarm, several time–history plots were generated. These include the agent spacing error, object pose tracking, and average inter-agent distances.

Spacing consistency is shown in Fig. 6. The caging potential successfully regulates the inter-agent distances once the formation forms.

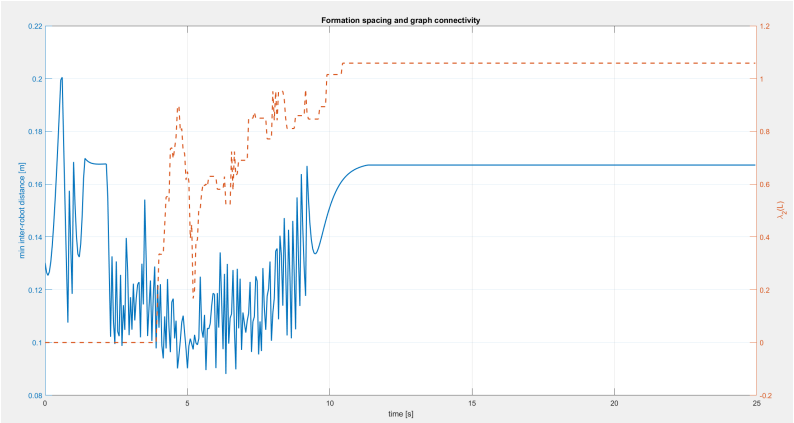

Figure 6: Inter-agent spacing error over time, demonstrating stable caging behavior.

Finally, Fig. 7 shows the distance between the robots and the object boundary. After initial transients, the robots converge to and maintain a consistent offset, indicating proper use of the object–avoidance potential.

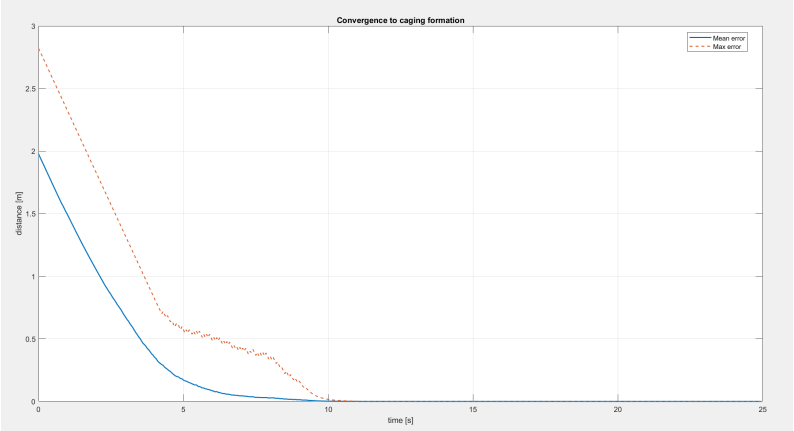

Figure 7: Robot-to-object boundary distance over time.

Discussion of Results

Overall, the simulation performs as intended. The robots reliably form a cage around the object without entering the interior, and the spacing remains uniform throughout the experiment. Once the object begins moving, the swarm successfully tracks its motion while maintaining the caging shape. The improved plotting settings (larger workspace, smaller robot markers, and corrected consensus graph) make the formation behavior much clearer than earlier iterations.

The main strengths of this approach are its simplicity and decentralization: each robot uses only local information and a lightweight analytic controller. The method scales well, is easy

to tune once initial gains are set, and produces smooth, stable motion.

There are also clear limitations. The object follows a prescribed trajectory rather than being physically pushed, so full force–closure and real manipulation dynamics are not yet demonstrated. The robots also assume perfect sensing and communication, which is optimistic for a real-world implementation. Still, the simulation shows that the core caging and tracking concepts are sound and provides a strong starting point for future work with more realistic physics.

V. Conclusion ⋆

In this paper, we successfully developed and outlined an algorithm for the autonomous and adaptable transport of any object in 2-dimensions using a homogeneous swarm of robotic agents. The transport problem was approached by adopting a caging strategy, which is mathematically simpler than optimising a generalised physics model and should lead to the same result. We used a generalizable Signed Distance Function (SDF), composed of the union of multiple circular SDFs, to represent the shape and pose of the transport object. A method was defined that can calculate evenly spaced caging points around the object's perimeter by first manipulating the SDF, running the marching squares algorithm, and then resampling in arc length. Finally, the object transport path was defined using a continuous, time-discretised coordinate transform. This transformation linearly interpolates the initial position and rotation to the final goal pose such that the robot swarm moves in unison while maintaining the caging formation. The resulting algorithm provides a foundation for the simple object transport task, a necessary component towards more general capabilities in smart manufacturing. Future research can expand on this work by extending this algorithm to consider more complex scenarios, including 3-dimensional transport, heterogeneous agents, obstacle avoidance, and, eventually, integrating it with machine learning and vision systems towards general object manipulation and construction.

References

[1] L. Chaimowicz, N. Michael, and V. Kumar, "Controlling swarms of robots using interpolated implicit functions," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2487–2492, 2005.

[2] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, 1992.

[3] M.-Y. A. Hsieh, S. G. Loizou, and V. Kumar, "Stabilization of multiple robots on stable orbits via local sensing," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2312–2317, 2007.

[4] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[5] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[6] H. G. Tanner, A. Jadbabaie, and G. J. Pappas, "Flocking in fixed and switching networks," *IEEE Transactions on Automatic Control*, vol. 52, no. 5, pp. 863–868, 2007.