EPiC Series in Computing
Volume 102, 2024, Pages 154–162
Proceedings of Conference on Cognitive and Computational Aspects of Situation Management 2023

Towards systems that dynamically change and evaluate abstractions

Ada Diaconescu[1], David King[2], Kirstie Bellman[3], Christopher Landauer[4], and Phyllis Nelson[5]

[1] Telecom Paris, IP Paris, LTCI, Paris, France
ada.diaconescu@telecom-paris.fr
[2] Air Force Institute of Technology, Dayton, Ohio, USA
davidking.jr@gmail.com
[3] Topcy House Consulting, Thousand Oaks, California, USA
bellmanhome@yahoo.com
[4] Topcy House Consulting, Thousand Oaks, California, USA
topcycal@gmail.com
[5] California State Polytechnic University Pomona, Pomona, California, USA
prnelson@cpp.edu

Abstract

Abstraction levels can be explicitly studied, changed and analyzed with metrics on their impact for a given problem. Also different methods for analyzing abstraction levels and their metrics can lead to different conclusions. Hence researchers usually iteratively experiment with these different methods to find the right abstraction level and metric for specific problems. To illustrate these points, we first study the use of Quad-Trees to characterize swarms, and then compare different methods using the metrics efficacy and efficiency. The goal of this work is to create an architecture and processes that will enable a self-aware system to conduct these types of experiments, and use these methods and metrics for analyzing the appropriateness and the impact of abstraction levels in order to improve its own performance.

1 Introduction and Background

One of the key aspects of Cognitive Situation Management (CogSIMA) is to have a system better perceive the situation, which includes the number and type of entities, the ongoing activities of the entities, and a characterization of the environment. The system here can be either autonomous or used to support a team of humans. Perception of the situation includes, of course, the scope of what is considered "part" of the relevant situation. The perception of the system also includes the scale and type of abstraction necessary for a given task in that environment. By "abstraction" we mean a description of the situation which contains a limited set of relevant characteristics. Such characteristics can be either observed directly or derived from observation. We are calling such changes in scope and abstraction "lenses" in this paper. Different tasks within even the same situation may be best accomplished with different lenses, so that a system must have a way to dynamically adjust its lenses, evaluate them for the task, and choose among them. Abstractions, of course, have been studied for a long time in psychology, in engineering and in computer science (e.g., object identification, tracking, swarms etc.) [9], and in physics (theories of the states of matter based on inter-particle forces [8]).

CogSIMA's development is partially rooted in the need to add perceptual and reasoning processes to systems which integrate diverse sensors for improving human situation awareness (SA). The definition of SA includes three segments: perception of the elements in the environment, comprehension of the situation, and projection of future status [3]. Situation Management (SM) brings in the control and actions that can result from SA. As Jakobson, et al. write: "We see SM as a framework of concepts, models and enabling technologies for recognizing, reasoning about, affecting on, and predicting situations that are happening or might happen in dynamic systems during pre-defined operational time. . . ." [7]. Humans are good at rapidly adjusting their abstraction levels based on what they are trying to accomplish. However, many computational systems still depend on humans to determine the right abstractions for the system at design time. Faced with novel and dynamic situations, it is critical that autonomous systems can rapidly adjust their use of abstractions, and that they can do this partially through processes of self-experimentation which determine the best abstractions to use. Also, even though humans are good at adjusting abstractions, they can fall prey to well-known problems in perception and in the use of abstraction. For example, a person can focus on specific aspects of the available data while ignoring others, or misinterpret information because of preconceived expectations [1]. Hence humans also need to explicitly test how well different abstractions and the methods for producing them fit the requirements of different tasks and environments.

2 Research Approach

2.1 Overview

The purpose of this study is to first illustrate the appropriateness of different lenses for answering particular questions about a targeted system and then to explore how these different lenses and methods can be evaluated via specific efficacy and efficiency metrics. Our targeted system is a computational simulation of swarm behavior. While our evaluation is manual, a similar process may be employed autonomously in the future to perform such evaluations and to select an appropriate lens.

2.2 Case study: swarm tracking

In this study, a computational process (observer) of a simulation tracks swarm formations and movements. This simulation emulates tracking natural swarms of fish, birds or bees; large animal migrations; human mobs; or technical systems such as drones or robots. We consider a set of entities, or agents, that move within an environment [11]. Entities avoid obstacles and interact with nearby entities to form groups that move together (i.e. swarms). Groups may merge with others or split into sub-groups. We focus on swarm location, employing three types of lenses and evaluating their suitability for this application. Experimental results (sec. 3) highlight which lenses are suitable in each case.

A lens is an observation perspective onto an observable system. It has an *input* (i.e., direct observations of the perceived system S), an *output* (i.e., processed perception produced for an observer O), and an *abstraction method*, transforming the input into the output. Several lenses may have the same input and output but different abstraction methods for mapping the former into the latter. An observer will use different lenses depending on their perceptual objectives. For instance, when the task is to estimate the number of entities in a swarm, an observer may employ either a lens that counts each individual – more accurate and more costly – or an alternative lens that estimates the swarm size by perceiving the swarm's average density and occupied surface – less accurate and cheaper [2]. To evaluate, and then select an appropriate lens, we start with two metrics: i) *efficacy* – can it answer the question correctly? (and if so, with what accuracy, probability of false positives/negatives, etc); and, ii) *efficiency* – what is the cost of producing the answer? "Cost" may include a variety of application-specific considerations such as the required computational resources, the time to complete the computation, and the need for additional information storage or transmission. In our example, we only focus on the size of the area that the observer must analyze, depending on each lens; this indirectly implies some of the other considerations above.

In developing the experiments, we defined many different lenses and methods for studying them. For example, at the level of observing the raw data from the swarm simulation, we created an Agent Counter (outputs the total number of agents in the simulation), Agent Positions (each Agent's position in X,Y coordinates within the simulation), Agent Directions (outputs each Agent's direction) and Agent History (outputs each Agent's position history over a chosen interval of time). Similarly, we defined methods for determining the number of swarms, the directions of swarms, whether Agents were identified with a swarm, whether swarms were splitting or joining, and where each swarm is located within the simulation space. We also defined methods for determining the centroids of swarms and their edges and perimeters in a variety of ways. To illustrate the approach taken, we next examine three methods: Quad-Tree, [5], [12]; QuadTree with associated QCircles; and Swarm Convex Polygon.

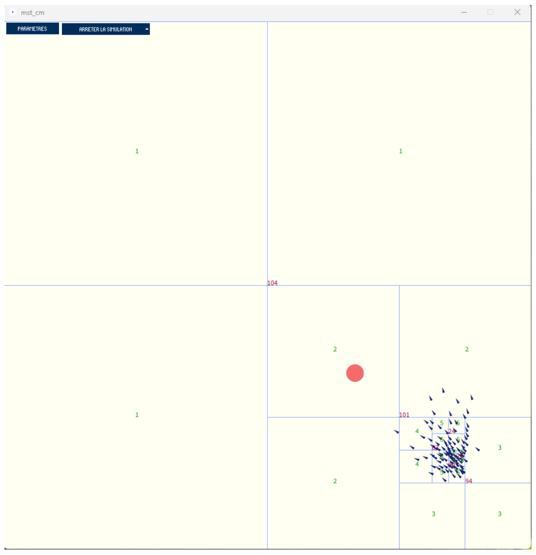
2.2.1 Quad-Trees:

We demonstrate locating swarms in the simulation using the quad tree data structure. We address such questions as: "Where are the swarms?"; "How many swarms exist?"; "What are the swarm trajectories?". If our snapshots are taken individually, without respect to time, then all notions of direction, movement, and potential reasons for swarm splits/merges are lost.

We define the score of a quad-tree node as shown in Eq. 1. The score is exclusively based on the node level and its children (if it has any) and not the exact number of agents. For a child node at level $qLevel_i$, the score equals $qLevel_i$. For each lower level ($qLevel_l, l < i$), the score is the sum of the level index l and of the scores of the four children of the node at that level $qLevel_l$. E.g., in Fig. 1b the score at $qLevel_4$ is 19 (printed in dark red at the center of the corresponding parent at $level_3$), which corresponds to the parent level 3 plus the scores of the 4 children (all leaf nodes at $level_4$).

$$qTreeNodeScore = \begin{cases} qLevel_i, \text{ if qTreeNode is a leaf node} \\ qLevel_i + qTreeScore_{NE} + qTreeScore_{NW} \\ \qquad\qquad + qTreeScore_{SW} + qTreeScore_{SE}, \text{ otherwise} \end{cases} \quad (1)$$

The swarm in Figure 1a has moved in a northwesterly direction and possibly split into two different swarms in Figure 1b, where the individual entities have been removed and only the quad trees are shown. The red dot in the figures represents an obstacle, and is the cause for the potential split in the original swarm. This figure highlights the power of using different lenses. The first lens shows the agent entities directly, which is easy for humans to process. The quad-tree may be more difficult for humans to interpret, yet may provide a simplified data set for machine evaluation.


(a) Quad tree representation of a boid swarm.

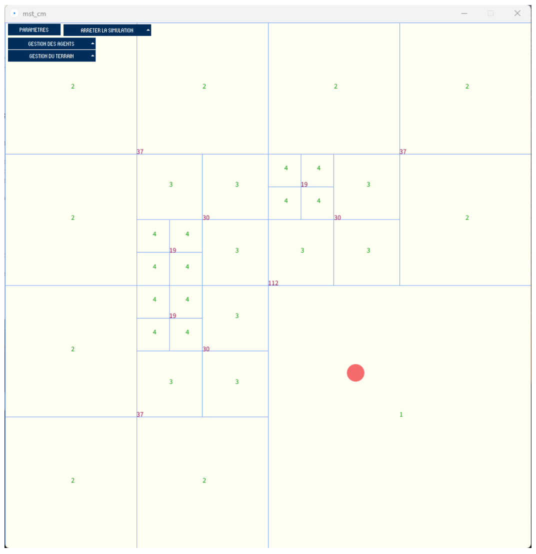

(b) Quad tree representations create information loss.

Figure 1: The Quad Tree lens begins by splitting the entire observed area into four initial quadrants. As entities enter a quadrant it is further subdivided into four smaller quadrants, as shown in (a). Splits occur in a quadrant if 9 or more entities are present, otherwise, the quadrant remains whole. In (b), at a later time, the entities are not displayed. An observer can no longer specify the exact number of entities in a quadrant. However, the observer can extrapolate, with a high likelihood, where the bulk of a swarm exists and, through observation over time, conclude the swarm's general direction.

Although there is information loss in the quad-tree lens, an observer can still make fairly accurate assessments of swarm location and general direction of movement. An observer may approximately locate where most agents are; although they may not tell if a swarm has just split/merged, or if two swarms just moved past each other without merging. The observer could deduce a splintering into other groups if the groups began moving in different directions, creating unique quad-tree values in different regions.

2.2.2 Quad-Circles:

Another lens based on quad-trees is called circular lenses, which are used to locate agents more accurately. The center of a circle lens represents the center of mass for a specific quad tree level. In Figure 2, the largest circles represent the center of mass for level one nodes in a quad tree. Medium circles represent level 2 nodes, and the smallest circles represent level 3 nodes. Each circle corresponds to a quad-tree node at a certain level. We use equations 2 to define the circle's center on the X and Y axes.

$$fraction_X = [(qScore_{NE} + qScore_{SE})/(qScore_{NW} + qScore_{SW})] * 2 \tag{2a}$$

$$delta = (qTree_{NE}.x - qTree_{NW}.x)/(fraction_X + 1) \tag{2b}$$

$$circle.x = qTree_{NE}.x - delta \text{ if East side of node has a higher score than West side} \tag{2c}$$

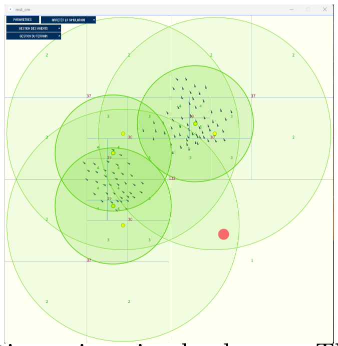


Figure 2: Quad tree representation using circular lenses. The center of circle represents the center of mass for a specific quad tree level. These circles create bounded areas where swarm members reside and represent movement and location more clearly than quad tree numbers alone. Again, the red circle represents an obstacle in the environment.

Once the circle's center is defined, we calculate the circle's radius as in Eq. 3.

$$circle.radius = \sqrt{qTree.width^2 + qTree.height^2} \tag{3}$$

The circles create a finer level of granularity than quad tree section numbers alone. There is still information loss, yet, in this case, the information loss comes with an observational gain. Namely, by tracking the circles without the quad tree splits, one can deduce direction, speed, and potential swarm splits and merges depending on the quad tree level selected. The circle lens benefits both human and machine observers. Human observers regain the visual cues for movement, direction, and swarm location. Machine observers can use the center of mass calculations to mathematically extrapolate these same tasks without the need for visual processing. Indicators for direction and speed based on such calculations could also be added to a display to assist humans.

2.2.3 Convex Polygons:

A ConvexPolygon lens was implemented by first identifying the swarms and then encapsulating each swarm in a convex hull. In two dimensions, the convex hull is the path that would be followed by a thread wrapping the swarm agents at each time step.[4]

2.3 Evaluation Scores

As an example of the use of metrics, we utilize the following normalized measures for efficacy and efficiency.

$$Efficacy_i = \frac{B_i}{B} \tag{4a}$$

$$E_i = \frac{B_i}{A_i} \times 100000 \tag{4b}$$

$$Efficiency_i = \frac{E_i}{E_i + 10} \tag{4c}$$

$$EvaluationScore_i = Efficacy_i \times Efficiency_i \tag{4d}$$

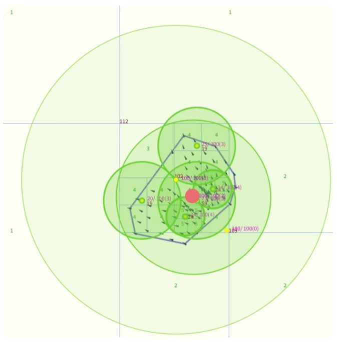
where B is the the total number of agents and B_i is the number of agents detected in the region (e.g., qCircle), and A_i is the total area of regions at each scale.[1]

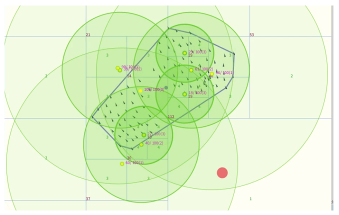
3 Illustrative Experiment

We carried out several experiments for answering the question, “Where are the swarms?”, using a swarm-formation and tracking simulation on an agent-based modelling platform in Java [6]. The simulation includes moving agents (boids) [11] and static obstacles placed within a 2D environment. Agents perceive other agents and obstacles within a configurable observation radius A ‘classic’ swarm-formation algorithm is employed where agents interact with agents that they can perceive via an attraction force, a repulsion force, and a force that aligns agents’ movements with that of near neighbors. Agent interactions lead to the formation of groups that travel together. Agents adjust their trajectories to avoid obstacles.

3.1 Simulation and Results

We used two types of lenses: QuadTree with associated QCircles and Swarm Convex Perimeter (Fig. 3). For the QCircle lenses, we tested seven levels, with increasing resolutions: from the largest area at $level_0$ (corresponding to the root node of the quad-tree) to the smallest area at $level_6$ (corresponding to the leaf nodes in the quad-tree). (Levels beyond 6 were found not to be useful and are omitted.) Lenses are adjusted automatically at runtime, depending on the swarm formation and position in the simulation space. The experiment was initiated ($t = 0$) with $N = 100$ agents, all placed in the lower right corner of a $900x900$ pixel 2D simulation space, resulting in a single swarm moving towards the upper left corner of the simulation space. Time was simulated in discrete steps, with the experiment lasting for 1000 steps. As time progresses, the initial swarm crosses an obstacle (red disc in Fig. 3a, $t = 232$), which the agents attempt to avoid by turning left and right. This causes the swarm to enlarge its area (Fig. 3b, $t = 413$) and finally to split into two different swarms (Fig. 3c, $t = 416$). The two swarms persist and move independently for the rest of the simulation (until $t = 1000$).


(a) Single swarm at $t = 232$


(b) Single swarm at $t = 413$

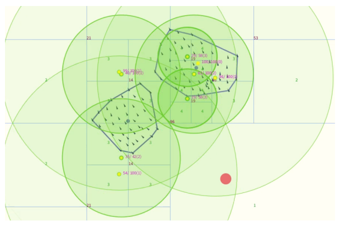

(c) Two swarms at $t = 416$

Figure 3: Swarm simulation example with two types of lenses: QCircles (green circles with darker lower levels and lighter higher levels) and ConvexPolygons (blue swarm perimeters)

[1]The intersection area of overlapping qCircles is counted multiple times. Also, the areas of qCircles that that extend beyond the simulation area are counted as if they were included within the simulation area.

Fig. 4 depicts the efficacy measures over the 1000 simulation steps for the different lenses used. The ConvexPolygon lens always produces $efficacy = 1$ as it locates all the agents (by definition). When using the QCircle lenses, the efficacy depends on the lens size $level_i$. Namely, larger lenses, such as $level_0$, $level_1$ and $level_2$ (with few exceptions) produce $efficacy_0 = efficacy_1 = efficacy_2 = 1$ most times, by including all agents within the corresponding QCircles. Conversely, smaller QCircles at $level_3$ fail to locate some of the agents – hence their efficacy varies between $efficacy_3 = 0.7..1$ during most simulation; and drops as low as 0.4 when the initial swarm becomes looser and splits into two smaller swarms (around $t = 410..490$). Lenses with even smaller areas, i.e., $level_4$ and $level_5$, perform relatively poorly, with $efficacy_5 < 0.7$ at most times, and even reaching $efficacy_5 = 0$ when the swarms become too loose to perceive any agent at all. The smallest lens at $level_6$ has $efficacy_6 = 0$ most times. In Fig. 3a, 3b and 3c all agents are included into the QCircles at $level_1$ and $level_2$ (cf. circles with two largest areas); yet lenses at $level_3$ leave out a few agents; while lenses at $level_4$ (only visible in Fig. 3a) merely capture agents located in the densest swarm areas.

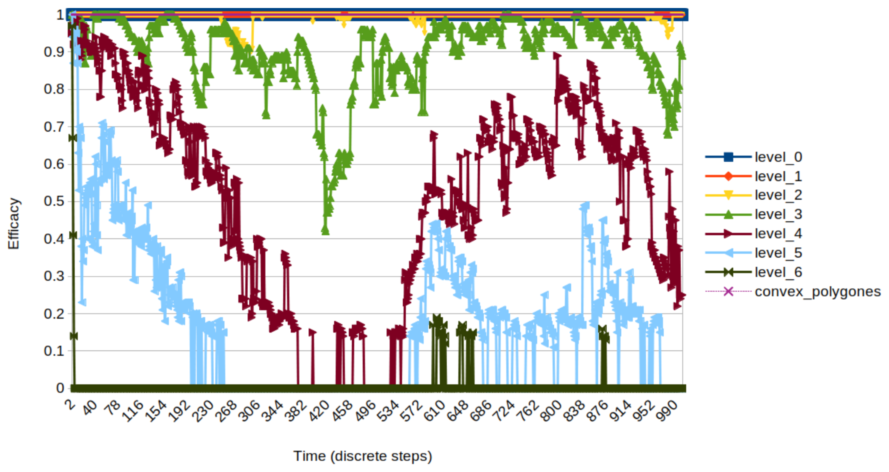


Figure 4: Efficacy of QCircle Lenses (levels 0..6) and Convex Polygone Lens (purple horizontal line at the top) – calculated as the proportion of agents located within this lens out of the total number of agents in the simulation

Fig. 5 depicts the evaluation scores for the applied lenses, over the 1000 simulation steps. This evaluation combines both measures of efficacy (i.e., how many agents were localised out of the total) and of efficiency (i.e., what is the area of the lenses used). Evaluation scores are normalised between 0 and 1, with 1 being the highest score (i.e., all agents are localised with high efficiency), and 0 the lowest score (i.e., few or no agents are localised and/or localisation is done inefficiently). The thick purple line represents the evaluation score for the ConvexPolygon lenses. The evaluation score here starts ($t = 0$) with the maximum value of 1, as all agents are situated in the same spot – hence facilitating their complete localisation with a minimum-area lens. As the simulation progresses the evaluation score varies between 0.80 and 0.95. As the ConvexPolygon lens always localises all agents, by definition, the efficacy of this lens is always 1 (maximum). The efficiency depends on the area of the ConvexPolygon lens, at each step. Hence, denser swarms will be located with higher efficiency (smaller polygon area) and looser swarms with slightly lower efficiency (larger areas). This explains the decrease of the evaluation score between $t = 0$ and $t = 413$, as the initial swarm that was very dense at $t = 0$ loosens progressively while going around the obstacle (Fig. 3a) and includes an area with very low density at $t = 413$ (Fig. 3b). As the swarm splits into two at $t = 416$ (Fig. 3c), it is located by two different ConvexPolygon lenses, with more compact areas overall. This is reflected in the evaluation score that features a small upwards jump at $t = 416$ (cf. Fig. 5); then increases progressively until $t = 600$ as the two swarms become tighter; then become looser again for the rest of the simulation. The other curves in Fig. 5 represent evaluation scores for the QCircle lenses at 7 different levels. $Level_0$ (dark blue line) features a constant evaluation score of 0.44. This is because both the efficacy and the efficiency of this lens are constant. As all agents are localised fully at all times, then $efficacy_0 = 1$. As the area of the QCircles at $level_0$ is always 1,272,345 pixels, including most of the simulation space (810,000 pixels), then $efficiency_0 = 0.44$. This means that this lens always localises all agents, yet with low accuracy.

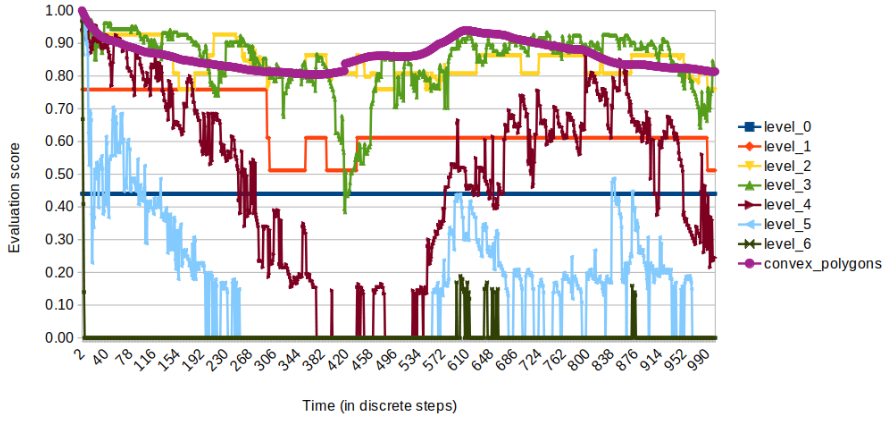


Figure 5: Evaluation scores for QCircle Lenses (levels 0..6) and Convex Polygon Lens (thick purple line)

4 Discussion and Future Challenges

We studied methods to determine and examine abstraction levels. As an example, we focused on a simulation of swarming. Experiments tested and evaluated two kinds of lenses, Quad Circles and Convex Polygons, for the question "where are the agents?" We exemplify an evaluation formula for the lenses' efficacy and efficiency. The results show how, based on these evaluation scores, an observer could rank and select the best lens in each situation to answer the given question. This example provides a possible basis for procedures that would allow an observer-system to evaluate and select lenses automatically at runtime. However, we emphasize that there are many potential methods and metrics to explore to determine the best lenses for specific cases.

To autonomously and successfully adapt levels of abstractions requires domain knowledge to decide on the right questions within a given context, and also meta-knowledge of the different methods that determine and analyze the abstractions. Trade-offs between the costs of using a given level of abstraction or analysis method and the sufficiency of any results can be particularly challenging. The knowledge required is not just about the methods, the metrics, and the domain: the system also needs to know about its own capabilities, which may change over time. For example, some of us created the CARS test bed[10] in which a heterogeneous group of robotic cars experimented to determine their own turning radii, sensor range, etc. These learned parameters were used to determine their choice of actions for a given task. In the case of abstraction selection, such self-knowledge might include how fast that system can process or how much memory it has available to perform a given algorithm.

Future work will deal with several challenges. First, in our simulation all the agents moved at the same speed, which is not necessarily representative of real world applications. Second, we did not deal with the challenges of occasional missing or bad data. We are also intending to experiment with how the movement in non-swarming agents affects the detection of swarms. Lastly, we intend to explore non-visual problems to study very different types of abstractions.

References

[1] Mica R. Endsley Cheryl Bolstad, A. M. Costello. Bad situation awareness designs: What went wrong and why. In *International Ergonomics Association (IEA) 16th World Congress*, Maastricht, Netherlands, August 2006. International Ergonomics Association.

[2] Ada Diaconescu, David King, Kirstie Bellman, Christopher Landauer, and Phyllis Nelson. Lensing: It is all about perspective. In *2022 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C)*, pages 119–125, virtual, 2022. IEEE.

[3] M. R. Endsley and D. Jones. *Designing for situation awareness: An approach to human-centered design.* Taylor & Francis, London, 2012.

[4] Jeff Erickson. Convex hulls, Retrieved 23 August 2023.

[5] Raphael Finkel and Jon Louis Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Informatica*, 4(1):1–9, March 1974.

[6] Processing Foundation. Welcome to processing!, Retrieved 06/22/2023.

[7] L. Lewis G. Jakobson, J. Buford. Situation management: Basic concepts and approaches. In K.V. Korolenko V.V. Popovich, M. Schrenk, editor, *Information Fusion and Geographic Information Systems.* Springer, Berlin, Heidelberg, 2007.

[8] David Goodstein. *States of Matter.* Prentice Hall, Englewood Cliffs, NJ, 1975. A basic review can be found in chapter 4.

[9] P. R. Lewis. Self-aware computing systems: From psychology to engineering. In *Design, Automation & Test in Europe Conference & Exhibition*, pages 1044–1049, Lausanne, Switzerland, March 2017. IEEE.

[10] Phyllis R. Nelson. Cars: A wrappings-based test bed for self* cyber-physical systems and their integration. In *2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, pages 44–48, Umeå, Sweden, June 2019. IEEE.

[11] Craig W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. In *SIGGRAPH '87'*, pages 25–34, Anaheim, CA, July 1987. ACM.

[12] Hanan Samet. *Design and Analysis of Spatial Data Structures.* Addison-Wesley, Boston, MA, 1989.