Structural Agent-Based Approach for 3D Geological Surface Modelling

ROGER NIYONGIRA

Thesis submitted to the University of Ottawa
in partial Fulfillment of the requirements for the
Degree of Master of Applied Sciences in Civil Engineering

Department of Civil Engineering
Faculty of Engineering
University of Ottawa

Abstract

Geological surface modelling is the process of creating 3D digital representations of the Earth's subsurface structures and surfaces based on geological, geophysical, and geospatial data. It's commonly used in geology, mining, petroleum exploration, hydrology, and geotechnical engineering. Traditional interpolation methods often rely on assumptions like spatial stationarity and high data availability, which are not always the case in earth sciences applications. These methods fail to capture local variation in geologically complex and data sparse areas, as they tend to be biased towards global means. This research explores how agent-based modelling (ABM) integrated with statistical and domain-specific knowledge can be used to improve geospatial estimation when data is sparse and incomplete, especially in geological surface modelling. In this study, structural agents are introduced as independent entities that interact with each other using rules from the classic BOID flocking algorithm: cohesion, separation and alignment. Each agent is defined by its spatial position (X,Y,Z), a normal (N_1,N_2,N_3) and a velocity (V_1,V_2,V_3) orientations, and updates its behaviour over time through local communication. A two-phase approach was used: Phase I tested if agents could self-organize, and Phase II evaluated how well agents could propagate information from real data points. The long-term goal of the study is to establish viability of a method for reconstructing surfaces of complex geological features such as faults, poly-deformed folds and regional fault networks. In the short term, this work focuses on determining what data configurations and agent behaviours support the simplest reconstruction scenarios, before addressing more complex geological cases.

Co-Authorship

The research presented in this thesis was conducted in collaboration with the following co-authors:

- Ousmane Seidou: Provided overall supervision, guidance on research methodology, and critical feedback throughout the study.
- Eric A. de Kemp: Contributed expertise in geological modelling, guidance on the structural agent-based modelling framework, preparation of validation data, and interpretation of geological results.
- Michael Hillier: Contributed expertise in computational methods, and constructive feedback on model development and results interpretation.

Acknowledgements

This study was funded by Natural Resources Canada (NRCan) through the Research Affiliate Program (RAP). The project is part of the Geological Survey of Canada's National Fault Inventory (NFI) initiative, contributing to the objectives of the Critical Minerals Geoscience and Data Program (CMGD). The financial and operational support provided by these initiatives is greatly acknowledged.

I would like to express my sincere gratitude to my supervisor, Dr. Ousmane Seidou, and my co-supervisor, Dr. Eric de Kemp, for their mentorship, insightful feedback, and unwavering support, which have shaped both this research and my professional growth. I am also grateful to the members of Dr. Seidou's research group, whose guidance, collaboration, and constructive discussions contributed greatly to this work. I also thank my colleagues at the Geological Survey of Canada for their support, collaboration, and for fostering a stimulating research environment throughout this project. In particular, I would like to thank Dr. Micheal Hillier for his valuable contributions, assistance, and the many insightful discussions that enriched this work.

To my friends and family, your encouragement, patience, and continuous support have been a constant source of motivation, and I could not have reached this milestone without you.

Table of Contents

List of Figures

List of Tables

List of Acronyms

Acronym	Definition
3-D	Three-dimension space
ABM	Agent-Based Modelling
CSV	Comma-Separated Values
DEM	Digital Elevation Model
FB8	Fisher-Bingham 8-parameter distribution
FB5	Kent distribution (5-parameter Fisher–Bingham distribution)
GIS	Geographical Information System
GUI	Graphical User Interface
IDW	Inverse Distance Weighted
PCA	Principal Component Analysis
PSO	Particle Swarm Optimisation
RBF	Radial Basis Function
SA	Structural Agent
SLERP	Spherical Linear Interpolation
VTK	Visualisation Toolkit
WCSB	Western Canada Sedimentary Basin

List of Symbols

N	number of agents in the simulation
P_i, P_j	positions of agent i and agent j
v	velocity vector of an agent
n	agent's normal vector (orientation)
r_{percep}	perception radius
r_{sep}	separation radius
F_{ij}	pairwise force between agents i and j
F_{total}	net force acting on an agent
w_{coh}	weight for cohesion
w_{sep}	weight for separation
w_{align}	weight for alignment
M	mean direction (Kent/FB5 distribution)
γ_1, γ_2, γ_3	orthogonal axes of Kent distribution
K	concentration parameter (Kent/FB5 distribution)
B	ellipticity parameter (Kent/FB5 distribution)
C	covariance matrix of neighbourhood normals (PCA)
λ_1, λ_2, λ_3	eigenvalues (PCA)
P_1, P_2, P_3	eigenvectors (principal directions from PCA)
q	quaternion rotation
t	interpolation factor in SLERP
c_1, c_2	cognitive and social coefficients (PSO)
	Strike and dip symbol: Horizontal line indicates strike, arrow shows dip direction

Chapter 1 – Introduction

Geological surface modelling has become critical as it connects scientific understanding of the Earth with urgent societal needs such as climate resilience, energy transition, water security, safe infrastructures, and environmental protection. These models are not only essential for advancing scientific knowledge but also serve as practical tools to support decision-making. They make it possible to anticipate and mitigate risks posed by natural hazards, to identify and manage subsurface resources including groundwater and hydrocarbon resources, and to guide the sustainable extraction of minerals and energy (Cao et al., 2024). Ultimately, geological surface modelling reduces exploration costs (De Kemp, 2007) while optimising resource allocations for large-scale infrastructure and development projects (American Geosciences Institute, 2025).

Geological models provide informative maps and visualisations that have proven valuable in a wide range of applications, specifically in identifying and characterizing geological faults and subsurface geometries. A fault is a fracture or a zone of fractures within the Earth's crust along which blocks of rock have moved relative to one another (U.S. Geological Survey, 2025). These movements are directly responsible for earthquakes, as the sudden release of accumulated stress causes one block to slip past another. Earthquake disasters are responsible for devastating natural hazards worldwide, causing loss of life, displacement of communities, and severe economic disruptions. The 2015 Nepal earthquake alone resulted in about 9000 fatalities, destroying homes and causing economic losses estimated at nearly US $10 billion, close to half the country's GDP at the time (Paudel, 2025). In the United States, recent hazard assessments by USGS and FEMA estimate annualized earthquake losses at US $14.7 billion (U.S. Geological Survey; Federal Emergency Management Agency, 2023).

Beyond their seismic significance, faults also serve as conduits for ore-forming fluids, influencing where and how minerals precipitate while simultaneously controlling the spatial distribution of mineral deposits (Carranza, 2009; Peng et al., 2023). Furthermore, faults can act either as conduits or as barriers to fluid flow, and their characterisation is therefore critical in groundwater studies (Cook et al., 2022) as well as in clean

geothermal energy exploration (Daniilidis et al., 2021; Gudmundsson, 2022) and in hydrocarbon systems, where they influence migration pathways, and the identification of oil and gas reservoirs (Liu et al., 2024).

In addition, knowledge of fault locations and subsurface geometries is also fundamental in other engineering applications including stability assessments and material characterisation for dams, bridges, and tunnels (Lin et al., 2007; Testa et al., 2024), and in the detection and mitigation of potential landslides, where geological surveys employ monitoring and mapping techniques to anticipate slope failures (Auflič et al., 2023). A recent economic analysis highlighted the financial benefits of producing geological maps in the United States of America, estimating returns of $13.9–20.6 billion on approximately $1.99 billion invested from 1994 to 2019 (American Geosciences Institute, 2025).

Building these models comes, however, with challenges, particularly the need to improve accuracy to meet the growing human and societal demands. In earth sciences, models are built for objects or regions that cannot be fully observed, such as ancient rock layers buried underground or deep riverbeds, unlike engineering models that can be fully specified before construction. Because Earth models are only partially known through sparse and difficult sampling, reliable methods are needed to capture their geometry and behaviour, whether for tunnels, reservoirs, or mineral extraction. Geoscientists therefore apply geospatial estimation techniques to create models, but each one comes with its own limitations. This study explores an agent-based approach that integrates existing geospatial estimation methods to reconstruct natural surfaces from sparse data, a major limitation of traditional methods.

1.1 Geospatial Estimation

Geospatial estimation refers to the process of predicting missing or unknown geographic information using available data. It has become a key tool in many disciplines, allowing researchers, scientists and decision makers to study and understand spatial patterns, relationships and trends in areas where direct measurements are limited or unavailable. Practical constraints such as cost, time, accuracy of measuring tools and location inaccessibility often make it impossible to measure every location with high certainty. To

fill these gaps, scientists rely on geospatial estimation methods ranging from simple deterministic interpolation techniques (Burrough & McDonnell, 1998) to more complex geostatistical (Pyrcz & Deutsch, 2014) and AI driven computational models (Hillier et al., 2023).

Geospatial estimation approaches differ from some interpolation methods in that they explicitly account for spatial correlation between observations. These techniques assume each point to be independent, which can introduce bias when applied to spatial data where such independence is rarely true. In spatial contexts, nearby observations influence each other, with closer observation having more influence on the point of interest than those farther away, as explained by Waldo Tobler's first law of Geography, "*everything is related to everything else, but near things are more related that distant things*"(Tobler, 1970). This concept is known as spatial autocorrelation.

Deterministic methods such as Inverse Distance Weighted (IDW), Thiessen (Voronoi) polygons, and nearest neighbour provide straightforward ways to estimate values at unsampled locations. These methods rely on geometric relationships, giving higher importance to nearby observations. For example, IDW gives more influence to the closest observations with an assumption that spatial influence decays with distance. Similarly, Voronoi polygons divide the space into regions and assign the corresponding nearest sampled point to the region. Despite their simplicity, these deterministic methods serve as a first step in spatial analysis and can be useful when working with simple evenly distributed spatial data.

However, as spatial processes often show complex anisotropic behaviour, deterministic methods become limited to providing accurate and reliable estimates in such scenarios. Geostatistical approaches provide a way to better estimate a spatial location and also to quantify uncertainties while maintaining the dependences between nearby observations. Georges Matheron showed that spatial dependencies between data points could be modelled mathematically to improve the accuracy of predictions (Matheron, 1963). His work laid the foundation of modern geostatistics with the development of the variogram, which quantifies how data similarly decreases with distance, and Kriging, a statistical interpolation method that is still used in geospatial estimation today.

Geological surface modelling has traditionally relied on these methods to generate continuous surfaces by interpolating scattered observations into scalar fields. These scalar fields are used to define geological interfaces, such as faults, typically through implicit modelling approaches. While effective in some cases, it tends to generalize areas with dense data and ignore surface variations, resulting in unrealistic geometries (de Kemp, 2021).

More recently, the increasing availability of data has led to the adoption of machine learning (ML) techniques, which are capable of capturing complex, nonlinear trends within spatial datasets. Despite their growing use, ML models face significant challenges in geospatial applications with sparse data. These include overlooking spatial autocorrelation, limited generalisation in areas with high spatial variability, or lack of statistical assessment which is critical for evaluating model reliability (Koldasbayeva et al., 2024).

These traditional approaches heavily depend on the quality and quantity of data to reproduce realistic models. In addition, they often fail to incorporate geological knowledge and structural constraints, resulting in models with topological inconsistencies (Hillier et al., 2021).

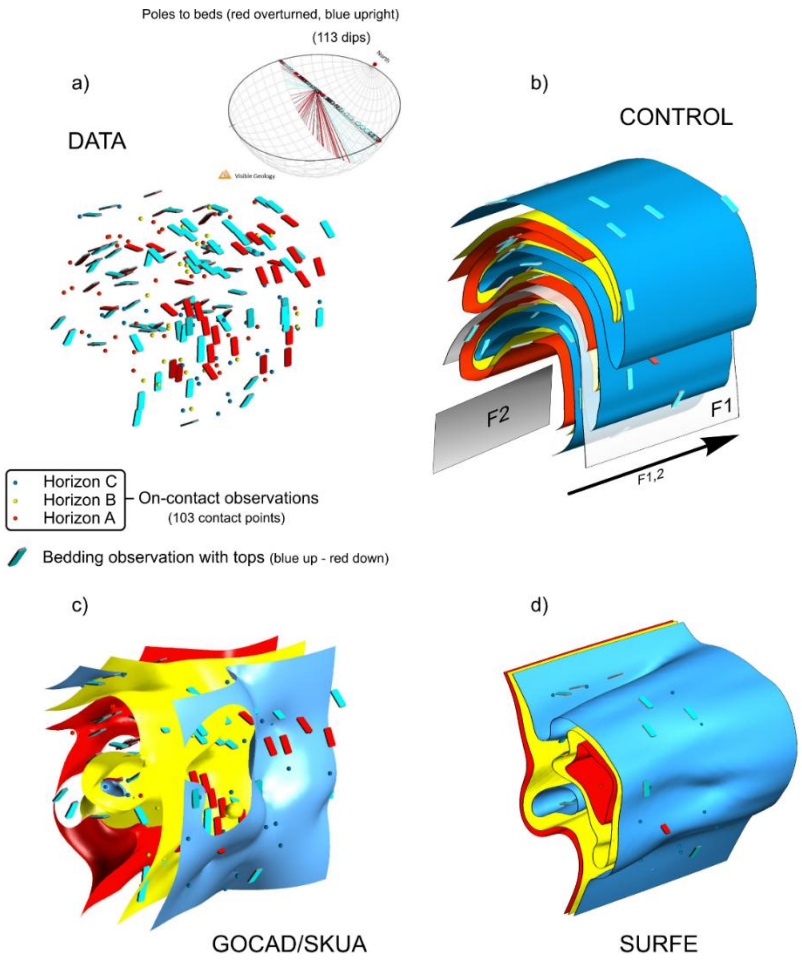

Figure 1: 3D surface model outputs showing: (a) uniaxial dip data, (b) control model developed with SPARSE (deKemp et al., 2004), (c) implicit surface models with GOCAD/SKUA (Jayr et al., 2008), and (d) surface model created with SURFE (Hillier et al., 2014). Reproduced from de Kemp (2021).

Agent-Based modelling (ABM) can help address these challenges in sparse data settings due to its ability to integrate traditional techniques with domain knowledge, in a decentralized approach. In ABM, an agent is a simple autonomous entity such as a point, particle, or object that follows a small set of behavioural rules. ABM is a modelling framework in which system behaviour emerges from interactions among many agents. When these agents interact with one another and with their environment, complex patterns can emerge from these simple rules, allowing the system to adapt locally rather than relying on a single global model. In spatial applications, ABM has been used to model dynamic systems such as land use change, traffic flow and population evolution,

capturing patterns that traditional models often miss. Its application in earth sciences, however, remains underexplored.

1.2 Problem Statement and rationale

Traditional geospatial estimation methods are used to predict unknown spatial values based on sampled observations. However, their performance heavily depends on the availability and quality of data. In Earth Sciences applications, data can be very limited due to high cost, terrain inaccessibility and the accuracy of measuring tools, especially when it comes to modelling subsurface features. In such conditions, traditional geospatial interpolation methods tend to fail to represent underlying patterns accurately. Deterministic methods ignore spatial variability while geostatistical techniques rely on spatial stationarity leading to bias toward the global means. In addition, these methods do not typically integrate domain-specific knowledge which results in inconsistent geological expectations. This is especially challenging in earth sciences, where the true output of the model is rarely known in advance. Unlike in many engineering disciplines, where models can be tested against clearly defined design specifications, geological models are often evaluated through judgement and consistency with known geological processes.

Therefore, there is a need for new estimation approaches that can model complex surfaces and better characterize natural objects at depth, while accommodating domain knowledge. This is particularly important in applications such as mineral exploration and riverbed modelling, where inaccurate surface representations can lead to costly misrepresentations.

1.3 Research Objective

The main objective of this research is to explore the potential of Agent-Based Modelling (ABM) as complementary tool to existing geospatial estimation methods for modelling complex geological surfaces when data is sparse. Specifically, the study aims to develop an ABM framework that, through a decentralized approach, reproduces geologically consistent surface structures. The method and resulting models will be validated visually against real and synthetic data to determine its capabilities and limitations. Furthermore, a

graphical user interface will be developed to allow model control, real-time parameter tuning, the selection of different types of data inputs, and the generation of triangulated surface meshes.

1.4 Overview and Thesis Structure

This study applies an agent-based approach to reconstruct geological surfaces given sparse and incomplete data. The main idea is to treat each structural agent as an independent and self-directed entity body carrying a small piece of information based on its local observations. The agent is given a spatial position, a direction (normal), and a velocity constrained to its own plane. From these agents' properties such as strike and dip can be derived for geological studies.

The simulation is initialized with only a few data points that have both known location and normal orientation obtained through field sampling, and some trace points with just spatial coordinates, typically derived from a digital elevation model (DEM) or incomplete field observations. First, using on-contact data with normals, the normal orientations for each of these trace points are estimated using the inverse distance weighted (IDW) technique, which provides a starting framework for the Agent-Based Model. Each of these data points then spawns an agent at a location near its own position, and a Kent distribution (Kent, 1982) is fitted from all nearby stopped agents and original data to obtain a normal orientation for the new spawns based on the local spherical distribution. This initialisation process helps to reduce the inconsistencies, making agents start with some directionally coherent orientations within their local neighbourhood, limiting their search for the best solution, unlike random initialisation approach. As soon as new agents enter the simulation, they start interacting with their neighbours by following the basic flocking rules of cohesion, separation and alignment with some spatial bias that influence agents to form a planar structure with their neighbours. For off-contact data with orientation, agents are directly spawned from these orientational data first, ignoring the trace location until one cohesive swarm has been created, where agents then migrate to trace locations. Agents' rotations are performed using quaternions to ensure a smoother rotation and avoid the gimbal lock effect. Since agents are updated in each simulation

timestep, a Spherical Linear Interpolation (SLERP) is applied to interpolate between rotations and limit abrupt rotation changes.

These techniques result in a propagation effect, where swarms of agents emerge from different regions determined by data locations. As the different swarms grow, they get in contact with each other and merge into one, while updating each swarm members' properties, as governed by the flocking rules.

The remainder of this paper is structured as follows: Chapter 2 reviews existing natural surface modelling approaches and explores the application of agent-based modelling in spatial sciences. Chapter 3 outlines the methodological framework, detailing the model development, agent interaction rules, and the statistical methods used to guide agent behaviour. Chapter 4 discusses simulation results, analyzing swarm behaviour and comparing the agent-based approach with conventional geological modelling approaches. Finally, Chapter 5 summarizes the key findings, discusses the limitations of the current work, and proposes directions for future research.

Chapter 2 – Literature Review

This chapter examines existing approaches to model natural surfaces (geological layers, terrain features, etc), from early interpolation techniques to explicit and implicit modelling, and more recent developments in machine learning. Furthermore, the proposed agent-based approach for reconstructing geological surfaces, along with relevant swarm technologies are reviewed.

2.1 Natural surface modelling

Natural surface modelling refers to the reconstruction of continuous geological or geomorphological features, such as stratigraphic layers, fault surfaces, or terrain topography. Unlike in many engineering disciplines, where a design is made with known specifications and clearly defined outcomes, natural surfaces are complex, irregular, and often only partially observable. In engineering applications, the object's final form is typically defined in advance and models are built to match a pre-determined goal. In contrast, natural surface modelling involves reconstructing surfaces that are formed through natural, often non-linear stochastic processes over long time scales (Wellmann & Caumon, 2018). This makes surface estimation uncertain and more dependent on limited and spatially variable observations, as well as expert interpretation.

Natural surfaces result from ongoing geological processes, influenced by environmental forces, structural deformations, and sedimentation patterns. Variables such as strike, dip, curvature, and continuity vary spatially, and often abruptly for example due to faulting, making modelling difficult in areas with sparse data. Several spatial interpolation techniques are applied to model these surfaces (Li & Heap, 2011) based on different data types including field measurements, digital elevation models (DEMs), cross-sections and seismic data. The challenge lies in capturing not just the model continuity, but also to preserve structural realism of surfaces as sometimes one is achieved at a cost of the other (Hillier et al., 2021).

2.1.1 Spatial interpolation methods

Spatial interpolation methods estimate values at unmeasured locations using a mathematical formula or statistical models on known data points, such as elevation or rainfall measurements. They are widely used in many applications that require spatial information, since data in the geosciences and environmental sciences are most often collected as discrete point measurements rather than as continuous surfaces (Li & Heap, 2014). Interpolation therefore provides a means to transform these scattered observations into continuous fields that can be visualized and analyzed. Such methods have been applied in the creation of digital elevation models (DEMs) (Chaplot et al., 2006), hydrology (Wagner et al., 2012; Zhou & Li, 2020), soil science (AbdelRahman et al., 2021), and geological modelling (Cui et al., 2025). In general, they can be divided into deterministic approaches, which rely on distance-based weighting functions, and geostatistical approaches, which model spatial correlation explicitly.

Among deterministic interpolation methods, Inverse Distance Weighting (IDW) is one of the simplest and most widely used. The origins of IDW can be traced to work in the late 1960s, where distance-based weighting functions were introduced to formalize interpolation from scattered observations (Shepard, 1968). Since then, IDW has become a standard component of many geographical information system (GIS) and geoscientific applications, owing to its conceptual simplicity and computational efficiency. The underlying assumption of the method is that the influence of a sample point diminishes with distance, such that nearby observations contribute more strongly to an estimated value than distant ones (Esri, n.d.). In IDW, the interpolated value $Z(x_0)$ at an unsampled location x_0 is expressed as a weighted average of the surrounding observations:

$$Z(x_0) = \frac{\sum_{i=1}^{N} w_i(x_0)\, Z(x_i)}{\sum_{i=1}^{N} w_i(x_0)} \quad (1)$$

Where,

$$w_i(x_0) = \frac{1}{d(x_0, x_i)^p}$$

$Z(x_i)$: observed value at location x_i,

$d(x_0,x_i)$: the distance between the prediction location and the sample point,
p : a power parameter that controls the rate of decay in influence with distance.

The selection of p, often set to 2 for the inverse distance squared weighted interpolation, varies depending on the applications and data type. Larger value of p place greater emphasis on nearer points, while smaller values allow distant observation to have more influence on the estimate. Barudžija et al. (2024) demonstrate that the optimal choice of the IDW power exponent depends on data density, where p of 1 or 2 yield best results for sparse datasets, while denser datasets benefit from higher values (3 or 4). Their analysis employs both visual pattern assessment and error metrics in a porosity mapping context. Figure 2.1 gives an illustration of how the weights varies with distances in IDW.

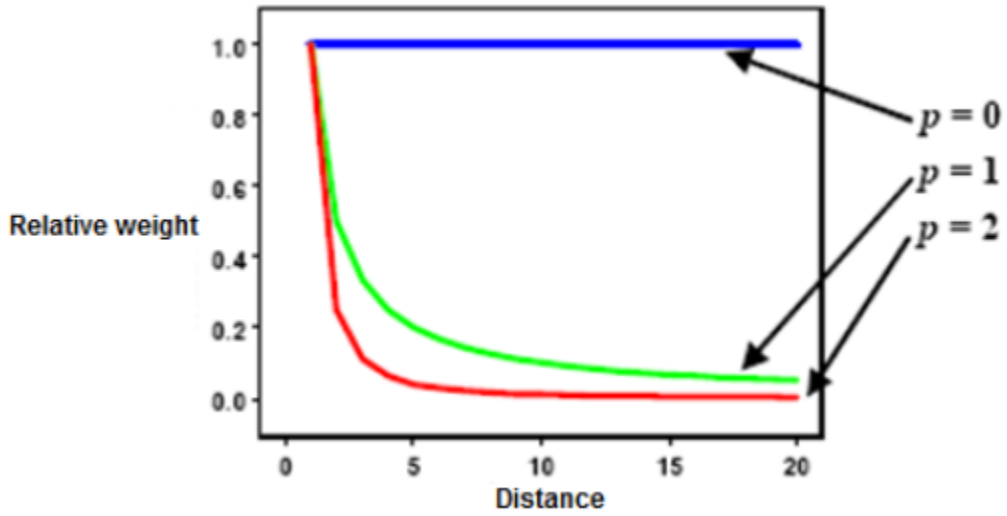

Figure 2.1 Variation of IDW weights with distance (Esri, n.d.)

Some approaches have suggested improvements to IDW applications. Wojciech (2020) proposed an optimisation of the method for creating digital terrain models by using a growing radius strategy, where neighbouring points are included dynamically until a specified number of samples is reached. Lu and Wong (2008) showed that an adaptive IDW in which the power parameter varies locally according to the spatial distribution of sample points can outperform the standard IDW in some cases. Other deterministic

methods such as radial basis functions and splines are also commonly applied; these are discussed in the explicit and implicit modelling approaches section.

While IDW and its variants remain popular due to their simplicity and ease of implementation, they share similar limitations. In particular, the method relies on user-defined parameters such as power exponent and search neighbourhood, which are often chosen arbitrarily rather than from the data (Lu & Wong, 2008). Moreover, IDW does not provide a way to quantify uncertainty of predictions (Goovaerts, 2000). Kriging, a geostatistical approach, provides a better way to both interpolate values at unsampled locations and to model the spatial correlation structure of data.

Kriging is one of the most popular geostatistical interpolation techniques, originally developed in the mining industry by the South African engineer Danie Krige in the 1950s, and later formalized through the theory of regionalized variables by Georges Matheron in the 1960s (Krige, 1951; Matheron, 1963). Unlike deterministic interpolations, Kriging treats the spatial phenomenon as a stochastic process with a defined covariance structure, allowing both prediction of values at unsampled locations and quantification of the associated uncertainty.

In its simplest form, Ordinary Kriging expresses the interpolated value at an unsampled location x_0 as a weighted linear combination of the surrounding sample values (Webster, 2005) :

$$Z(x_0) = \sum_{i=1}^{N} \lambda_i Z(x_i) \quad (2)$$

Where $Z(x_i)$ are the observed values at neighbouring locations, and λ_i are the kriging weights. These weights are derived by solving system of linear equations that minimizes the estimation variance.

Similar to IDW, several variants of kriging have been developed to suit different data conditions. The most common one Ordinary Kriging (OK), which assumes a constant but unknown mean and produces unbiased estimates by weighing nearby points according to their spatial correlation. Universal Kriging (UK) allows the mean to vary with location, modelling the residuals as a correlated random field. In cases where additional variables

related to the target are available, Cokriging (CK) incorporates this secondary information into the interpolation to improve accuracy (Hilal et al., 2024). Regression Kriging integrates regression models with spatial correlation in the residuals (Odeh et al., 1995).

Kriging has been widely applied across earth and environmental sciences, including digital terrain models (DTM) (Bernardes et al., 2006), soil moisture estimation (Anand et al., 2021), and ore grade estimation (Maniteja et al., 2025). However, due to its assumption of spatial stationarity and constant mean, the resulting predictions can be misleading in heterogeneous settings and in sparse or are non-stationary data cases (de Kemp, 2021; Luo et al., 2025).

2.1.2 Explicit modelling approaches

Explicit modelling consists of constructing surfaces by directly defining their shapes using features such as points, lines, and polygons. Among these, Bézier curves and surfaces, fundamental tools in computer-aided geometric design (CAGD), can be used to reconstruct natural surfaces. A Bézier curve is a parametric curve defined by a set of control points, where the curve is influenced by these points. This can then be extended into Bézier surfaces using tensor products (Farin, 2002). Several methods have been proposed to improve the classic Bézier curves and surfaces. For example, a generalized fractional Bézier rule surface can be constructed using the tensor product of the generalized fractional Bézier basis functions (Said Mad Zain et al., 2023).

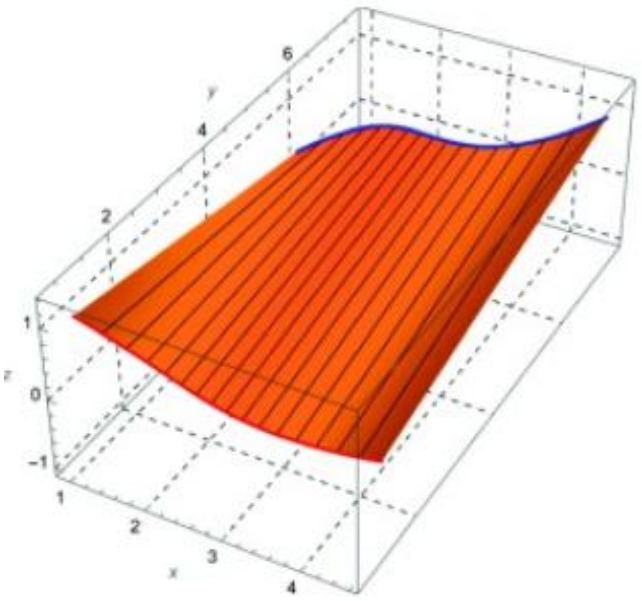

Figure 2.2: Surface from Bezier curves (Said Mad Zain et al., 2023)

In geological context, 3D Bézier construction tools have been applied to generate geological surfaces by shaping control points that define smooth curves and surfaces (de Kemp, 1999). In this approach, surface geometry is constructed from a network of Bézier curves, which are interpolated into continuous surfaces, as shown in Figure 2.3.

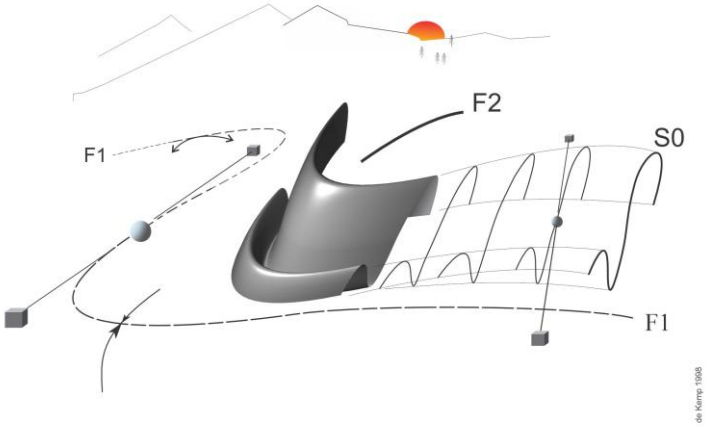

Figure 2.3 Geological surface from Bézier-controlled complex fold visualisation tool (de Kemp, 1999).

Another widely used class of explicit modelling techniques comprises parametric spline surfaces, such as B-spline (Fang & Hung, 2013) and non-uniform rational B-spline (NURBS) surfaces (Dimas & Briassoulis, 1999). Parametric spline surfaces build on a

grid of control points and knot vectors to define smooth, flexible geometry (Sprague & de Kemp, 2005). While they offer good smoothness, continuity and local manipulation, they are very dependent on how the knot vector is chosen, and poor parameterisation can introduce oscillations or instability in cases of sparse and unevenly distributed data (Fang & Hung, 2013). Furthermore, NURBS weights can add another layer of mathematical and implementation complexity (Piegl, 1991). Discrete smooth interpolation is also another commonly used explicit approach, with the goal of minimizing a global smoothness criterion across a triangulate mesh while respecting scattered data constraints (J.-L. Mallet, 1992).

These explicit modelling approaches can be good when model control for smooth surface is desired, but their requirement for manual effort can be time-consuming and prone to uncertainties on complex geometries. In addition, their scalability is limited (Hillier et al., 2023).

2.1.3 Implicit Modelling approaches

Implicit modelling approaches consist of reconstructing surfaces by treating them as level-sets (iso-surfaces) of a continuous three-dimensional scalar field s(x), fitted so that s(x) = 0 at observed contacts or structural measurements (Wellmann & Caumon, 2018). These methods allow automation, unlike explicit approaches, by representing surfaces with mathematical functions instead of manual polygon editing, which also reduces user biases (Beni et al., 2025). Radial basis functions RBF are used to improve the smoothness of continuous surfaces, but at a cost of increased computational requirements (Cuno et al., 2005).

Several studies have used variants of implicit modelling to estimate surfaces. A hybrid approach combining RBF and harmonic analysis was used to estimate and characterize fold geometry from field data (Grose et al., 2017). Implicit modelling has also been used to generate 3D models of stratigraphic formations from remote sensing and digital elevation models DEMs (Caumon et al., 2013). Traditional implicit modelling approaches have since been extended to more recent technologies, incorporating neural networks to improve modelling accuracy. In computer graphics, Schirmer et al. (2024) demonstrated that surfaces could be reconstructed with geometric implicit neural

representations (INRs) to approximate signed distance functions (SDFs) from point cloud or posted images. A different INR-based approach for geological applications, Geological Implicit Neutral Network (GeoINR), was introduced in a study to model stratigraphic surfaces while incorporating geological constraints such as unconformities and stratigraphic relationships (Hillier et al., 2023).

Commercial modelling platforms such as GoCAD (J. L. Mallet et al., 1989), Geomodeller (Carmichael & Ailleres, 2016), integrate implicit modelling engines into interactive interfaces allowing users to build and update 3D geological models in real time. Due to their simplicity in allowing automation, strength of implicit approaches, and model manipulation, these platforms have gained popularity in geological modelling. Despite their advanced capabilities in complex surface modelling when quality data is available, implicit modelling approaches face challenges when reconstructing surfaces from sparse data. In such scenarios, these method tend to over generalize dense data region resulting in the loss of a global geologically meaningful structure (de Kemp, 2021). They also fail to model complex structures such as fold geometries, usually explicitly defined by user instead of data approach, which can lead to human bias (Laurent et al., 2016).

2.2 Agent Based Modelling

Agent-based models (ABMs) are computational models in which agents can represent an entity, a person, or an object that can interact with each other. Agents collect information about changes within the system and make autonomous decisions based on defined rules and constraints. ABMs provide a way to model dynamic systems where interactions are complex to represent, by having an agent act on behalf of a small part of the system and carry or exchange information. Their application varies widely, from computer gaming (Mac Namee, 2012) to social sciences (Templeton et al., 2024), and urban and environmental planning (González-Méndez et al., 2021; Sousa et al., 2025). Elements of a typical agent-based model are illustrated in figure 2.4.

Figure 2.4: Components of an agent-based model (Turrell, 2016)

Due to their ability to propagate local information on a global scale, agents have been used in areas where traditional methods fall short. In a study conducted on a smart grid system, intelligent agents has shown that simulating a distributed control system rather than a centralized way can yield efficient demand side management (Malik & Lehtonen, 2016). In spatial contexts, ABMs have also proven their abilities to model social-spatial aspects, such as complex changes in food security projections (Hemerijckx et al., 2025), or impact of income-based residential preferences (Barros, 2012).

Although most ABMs simulate direct interactions without incorporating any topological or directional information, which is necessary in natural surface modelling, some studies have attempted to generate surface by employing the decentralized framework inherent to ABMs. For example, freeform architectural surfaces were constructed with an agent-based system by generating principal curvature strips on those surfaces (Chai et al., 2024). While these architectural applications demonstrate the potential of agent coordination for generating smooth engineered surfaces, they rely on well-defined geometric constraints and do not address the irregular, discontinuous nature of natural surfaces. In the geosciences, spatial agents were used in a novel approach, integrating structural information such as dip data with polarity and orientations, to generate a geological fold surface (de Kemp, 2021). Despite showing promising results for ABM in geosciences, the application of spatial structural agents remains in its early stage, having

been demonstrated on relatively simple surfaces, it did not extend to more complex geological surfaces with high topological variability. As shown in Figure 2.5 (a) spatial structural agents can self-organize cohesively based on control data and in Figure 2.5 (b) the red surface generated using these agents was more realistic than the yellow ellipsoid generated with implicit modelling approach (de Kemp, 2021).

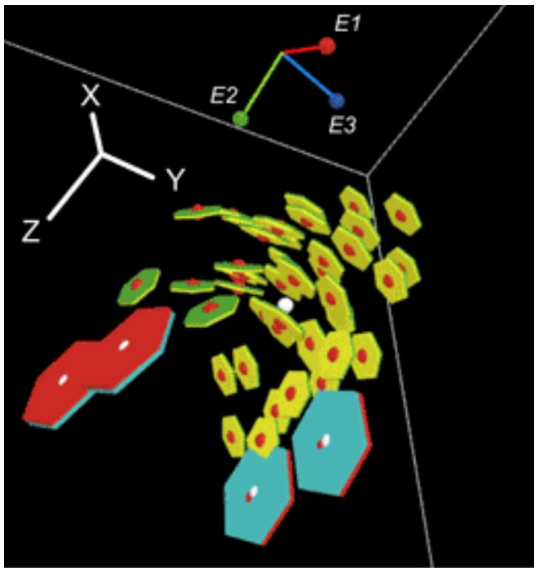

Figure 2.5 (a) : Example of spatial continuous structure with structural agents (de Kemp, 2021)

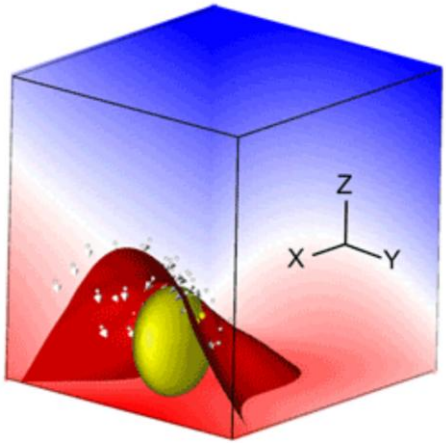

Figure 2.5 (b): comparison between structures from implicit model (yellow surface) and structural agents (red surface) (de Kemp, 2021)

2.3 Directional statistics

Directional statistics is a branch of statistics that focuses on the analysis of data representing directions, where observations are constrained to lie on a circle, sphere, or other manifold (Pewsey & García-Portugués, 2020). It has applications across many disciplines such as biology to measure animal directions (Fitak & Johnsen, 2017), in meteorology to analyze wind energy (Carta et al., 2008), and in geology to quantify 3D bedding orientations and fracture planes (Trauth, 2007). Unlike scalar variables, directional data require statistical methods that account for their periodicity and geometry, where standard statistics can produce inaccurate results. For example, treating circular observations as Euclidian observations, two opposite directions would sum to zero. In the case of axial data, where directions separated by 180^o are considered equivalent, data is first converted to circular data by doubling the angles before the analysis (Mardia & Jupp, 2010).

For directional datasets defined on the sphere, probability distributions such as Fisher-Bingham family provide flexible models to capture both concentration around a mean direction and elliptical patterns of dispersion on the sphere (Bingham, 1974). The Fisher Bingham 8-parameter (FB8) distribution, in particular, can model a wide range of directional patterns, including those with asymmetric spread not aligned with standard great-circle symmetry (Yuan, 2021). FB8 has been applied in different fields including biology to model protein structure (Boomsma et al., 2006) and wireless communication for multiple-antenna systems modelling (Alem et al., 2015). While FB8 offers great flexibility, its eight parameters can make estimations computationally intensive especially for large datasets (Paine et al., 2018). The Kent distribution (Kent, 1982), a special case of the FB8 with only five parameters, is a widely used alternative which helps to reduce the complexity of the model while retaining the elliptical dispersion on the sphere.

The Kent distribution is defined on the unit sphere and is parametrized by a mean direction vector μ, as major axis vector γ_1, and to concentration parameters κ and β. The parameter κ controls the overall concentration of data around the mean direction, similar to Fisher distribution, while β introduces elliptical anisotropy, controlling how dispersion differs along the major (γ_2) and minor (γ_3) axes orthogonal to μ.

The probability density function of the Kent distribution is given by:

$$f(x) = c(\kappa, \beta) exp\{\kappa \mu^T x + \beta[(\gamma_2{}^T x)^2 - (\gamma_3{}^T x)^2]\} \quad (3)$$

Where x is the point on the unit sphere and $c(\kappa, \beta)$ is the normalisation constant.

In geological modelling, these parameters can be used to analyze the structural form and dispersion based on orientational data. In a study on the Purcel Anticlinorium of southeastern British Columbia, bedding plane orientations were fitted with a Kent distribution, yielding an ellipticity ratio $2\beta/\kappa = 1$, suggesting a cylindrical fold structure (Hillier et al., 2021).

2.4 Swarm intelligence

Swarm intelligence (SI) refers to collective behaviour of decentralized, self-organizing systems, typically involving many simple agents interacting locally with each other and with their environment (Fleischer, 2005). In most practical applications, these systems still require manual control, making the term intelligence somewhat ambiguous, as definitions of self-organisation vary across different applications (Gershenson et al., 2020).

SI has been applied to a broad range of algorithmic and physical systems. In computational contexts, one popular method is particle swarm optimisation (PSO), where a swarm of articles searches a multidimensional space to find the optimal solution (Sengupta et al., 2018). PSO has been applied in various optimisation problems such as path planning of underwater vehicle (Tang et al., 2010), renewable energy for finding the best trade-off between reliability and hybrid system costs (Anoune et al., 2018), and in microgrid management for optimising unit commitment under uncertainty in wind power, load forecasts, and electric vehicle controllability (Moghaddas Tafreshi et al., 2016). In PSO, particle velocities are updated based on their current velocity (inertia term), the displacement toward their best-known position (cognitive term) and the displacement toward the globally known position (social term) (Shi & Eberhart, 1998):

$$v(t+1) = wv + c_1 r_1 \odot (P^* - x) + c_2 r_2 \odot (g^* - x) \quad (4)$$

Where,

w = inertia weight, c_1 = cognitive weight, c_2 = social weight,

r_1, r_2 = random numbers to add stochasticity,

P* = personal best positions,

g* = global best position

x = positions,

v = velocities

$\odot$ = element-wise multiplication

Other common approaches of swarm intelligence include ant colony optimisation (ACO), inspired by real ants based on how ants find and mark their paths by depositing pheromones on the ground to guide other ants in the colony (Mamur et al., 2022). ACO has also shown success in areas such as transport route planning (Chandra Mohan & Baskaran, 2012), and effectively routing wireless sensor networks (Okdem & Karaboga, 2009).

In recent years, advances in real-time localisation and wireless communications have led SI to evolve into more physical and dynamic aerial systems. Among these are the unmanned aerial vehicles (UAVs) or drones that operate with minimal human intervention, allowing applications such as environmental monitoring (Brust & Strimbu, 2015) and surveillance ,where UAVs can detect, track and intercept intruders (Stasinchuk et al., 2021). In the entertainment industry, UAVs have also gained popularity in the way their ability to create aerial performance, especially for drone light shows, where thousands of drones synchronize to form dynamic, three-dimensional animations. A notable example is the 2024 Shenzhen light show, which deployed 7598 drones to create the larges aerial images (Zhan, 2024) showing the advancement of swarm intelligence.

Chapter 3 - Methodology

This study was conducted using structural agents characterized by three attributes: a normal orientation, a velocity, and a spatial position. To visually distinguish structural continuity from irregularities, the agent was designed, in Blender, an open-source 3D modelling software, as a hexagonal prism inspired by the spatial agent design proposed by de Kemp (2021). The prism is given a blue top and red bottom to show the polarity of its normal vector. The agent is always oriented according to its normal orientation, with its velocity constrained to be perpendicular to the normal orientation, thus lying on its plane.

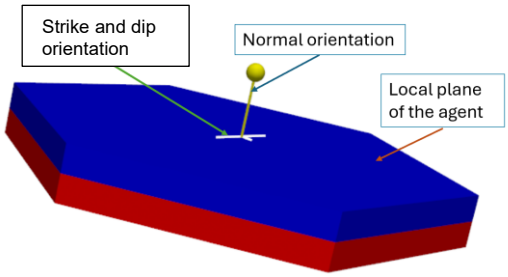

Figure 3.1 Structural agent

The experiment was completed in two main phases. In the first phase, agents interacted with one another to assess whether local communication could yield a cohesive planar structure. In the second phase, real geological data points were introduced, requiring agents to adapt their behaviour to sampled sparse data in order to interpolate and extrapolate missing information. For access to code used in this study, see: https://github.com/Roger-Niyongira/Structural-agent-based-model

3.1 Phase I: Agent communication

During this phase, agent behaviour was designed to test whether a swarm of agents, interacting with one another, could self-organize into a geological planar feature. A population of N agents was initialized with a random position and velocity within a bounded 3D space. The initial normal orientation of each agent was derived from its

assigned velocity to ensure that each agent's motion remained constrained to its local plane. First, a global reference velocity vector along the X-axis (1, 0, 0) and reference normal along the Z- axis (0, 0, 1) were defined. The angular difference (α) and the axis of rotation between the reference velocity and each agent's randomly assigned velocity were then computed.

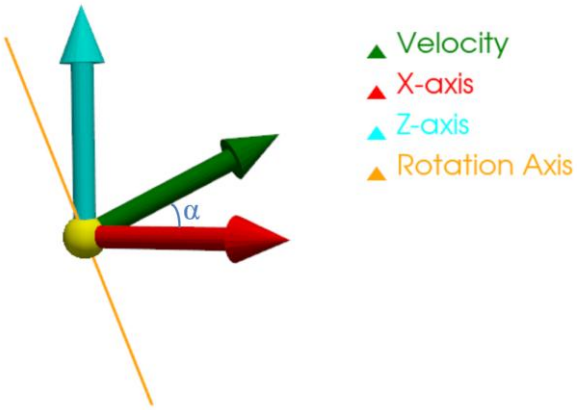

Figure 3. 2 (a) Rotation axis and angle α between reference velocity and agent velocity

The same rotation was applied to the reference normal using quaternion rotation, resulting in a normal vector, orthogonal to the velocity, for each agent.

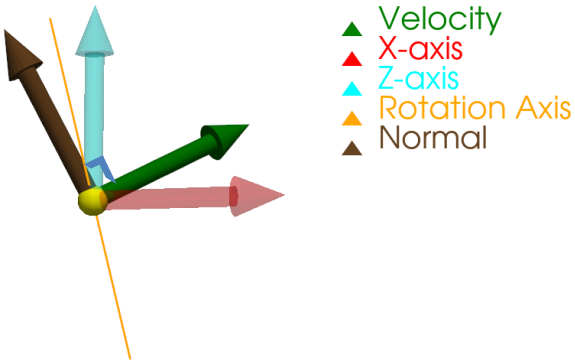

Figure 3.2 (b): Resulting agent's normal vector

This is applied to each individual agent to obtain initial states of agents

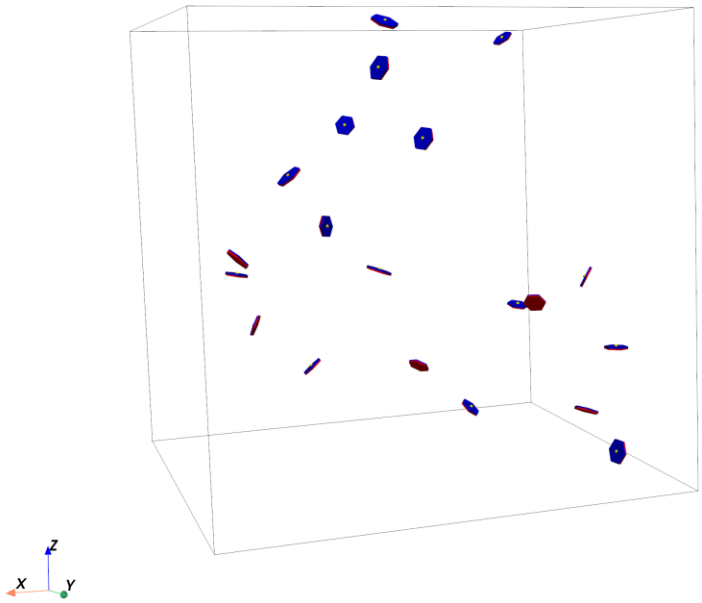

Figure 3.2 (c): Initial states of agents

Agents have three primary rules to follow based on the classic Boid flocking algorithm: separation, alignment and cohesion (Reynolds, 1987). Figure 3.3 illustrates a typical flocking scenario. The focal agent is shown in yellow, with its perception radius represented by the green circle. Agents within this radius (cyan) are considered neighbours for interaction. The centroid of these neighbours (red point) defines the cohesion target toward which the focal agent is steered. Around each neighbouring agent, a smaller red circle represents its separation zone, within which repulsive forces act to maintain an appropriate inter-agent distance and avoid collisions. Agents outside the perception radius (black) do not influence the focal agent's behaviour. The purple arrow indicates the average heading of the neighbours, used for alignment.

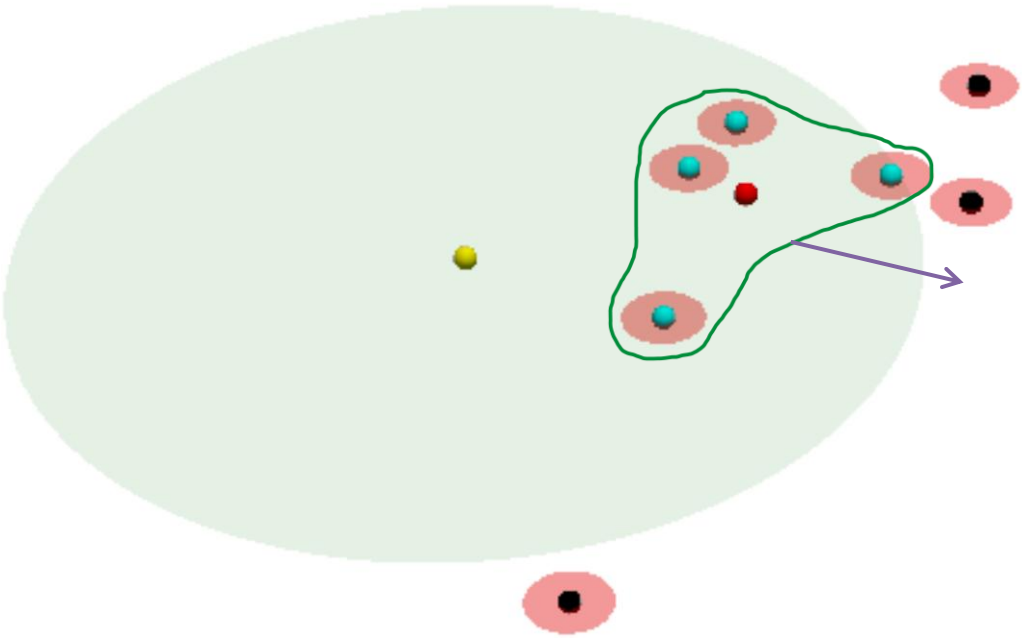

Figure 3.3: Flocking of agents.

In this simulation, the standard flocking rules have been modified to include physical properties such as mass and distance. Instead of having an agent head towards the mean position of its local flockmates to maintain cohesion, a force inspired from the Newton's law of universal gravitation was computed to simulate distance-based influence between agents. Newton's force is given by:

$$F = G\frac{m_1 * m_2}{r^2} \tag{5}$$

F: force

G: Gravitational constant ($G=6.67430 * 10^{-11}$) $\frac{N.m^2}{kg^2}$

m_1: mass of object 1

m_2: mass of object 2

r: distance between objects

All agents in the simulation are assigned identical features, including mass, which is held constant throughout the simulation. The simulation environment is spatially confined, thus resulting in a very small force when the gravitational constant G is applied. Therefore, the force applied between agents can be simplified as follows:

$$F = \frac{1}{r^2} * c \tag{6}$$

F: force
r: distance between agents
c: adjustable constant to control the strength of attraction and repulsion

The resulting force was then used to compute each agent's velocity that would respect the basic flocking rules, modified to introduce spatial bias:

3.1.1 Cohesion

This force was modelled as a gravitational-like attractive force, influencing each agent to join its local members, with nearest neighbours having a higher influence defined by the inverse distance. This approach makes each agent act like a particle attracting each other particle within its perception distance. This attractive force will be modulated by agent orientations creating an anisotropic attractive force mimicking a local flat world gravity field (see Section 3.1.3).

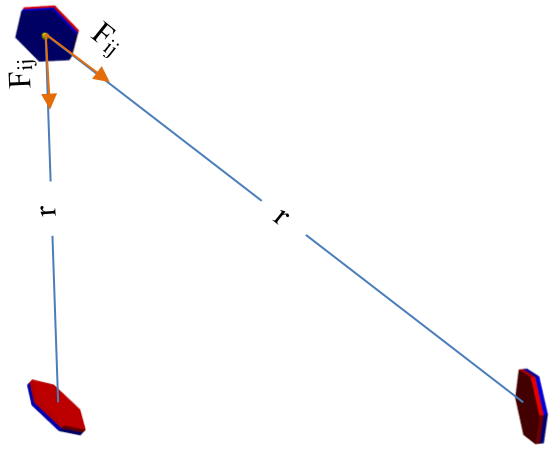

Figure 3.4: Agent's attraction

For each agent i, all neighbours j within the perception radius r_{percep} are gathered. The scalar pairwise attraction F_{ij} is given by Eq. 2 which is then projected along a unit direction vector from i to j. The net cohesion force for agent i is given by the sum of all contributing forces:

$$F_{cohesion,i} = \sum_{j \in N_i} F_{ij} \frac{P_j - P_i}{\|P_j - P_i\|} \tag{7}$$

Where:

$N_i = \{j : r < r_{percep}\}$: set of neighbours within the perception distance

P_i and P_j : positions of agents i and j, respectively,

F_{ij} : the scalar attraction from Eq. 6

$\frac{P_j - P_i}{\|P_j - P_i\|}$: normalized direction vector from agent i to agent j

3.1.2 Separation

To keep agents from crowding one another, a short-range separation distance was defined as r_{sep}. When another agent enters this range, a repulsive force is applied to push this agent away. This force is given by the negative of the pairwise attraction defined in Eq. (6).

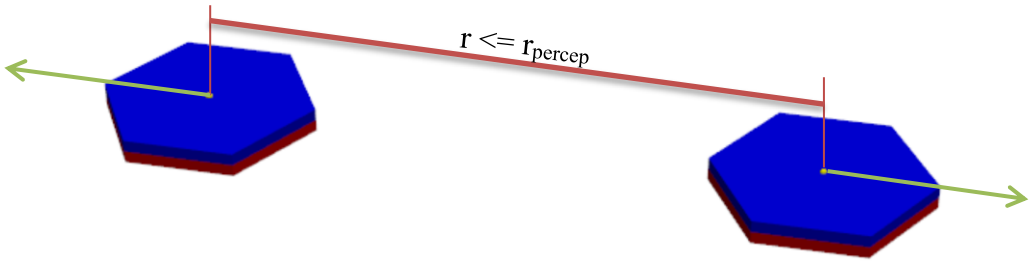

Figure 3.5: Agents' repulsion when they come within separation radius

Therefore, the separation force is given by the following formula:

$$F_{separation,i} = \sum_{j \in S_i} (-F_{ij}) \frac{P_j - P_i}{\|P_j - P_i\|} \quad (8)$$

Where:

$S_i = \{j : r < r_{sep}\}$: set of neighbours within the separation range of agent i

P_i and P_j : positions of agents i and j, respectively

F_{ij} : the scalar attraction from Eq. 6

$\frac{P_j - P_i}{\|P_j - P_i\|}$: normalized direction vector from agent i to agent j

3.1.3 Eigen based spatial biasing

To encourage agents to locally align with their neighbours, both in position and orientation, local eigenvectors were derived through eigen decomposition of the normal in each agent's neighbourhood. For each agent i, all neighbouring normal vectors within

the perception distance r_{percep} were collected. Using numpy library in python, a covariance matrix (C) of these normal was computed, which calculates how each normal component (N_1,N_2,N_3) varies relative to the others (see Appendix B for code details). The resulting matrix was then passed into the numpy function as an input (numpy.linalg.eig(C)) to perform eigen decomposition. This function returns:

- Eigenvalues: representing the variance along each axis.
- Eigenvectors: the directions associated with those variances, where P_1 is the direction of the greatest variance, P_2 the intermediate direction and P_3 the direction of the least variance (estimated local plane).

Each force vector F applied to the agent was decomposed along the three local eigenvectors (P_1, P_2, P_3), and each component is scaled by a corresponding directional weighting array $\alpha = [\alpha_1, \alpha_2, \alpha_3]$. This results in a modulated force that emphasizes a motion in specific directions:

$$F_{eigen} = \alpha_1 \,.\, (F.P_1)P_1 + \alpha_2 \,.\, (F.P_2)P_2 + \alpha_3 \,.\, (F.P_3)P_3 \qquad (9)$$

For cohesion, α is set to [0.1, 0.1, 0.9] by default, giving higher bias to align along P3, while for separation the opposite configuration was applied with α = [0.9, 0.9, 0.1], encouraging agents to spread laterally along the local plane. For this study, focused on structural continuity, these values are fixed but could be adapted for specific applications such as situations where local planar cohesion is not as important.

3.1.4 Alignment

After introducing spatial biasing to guide agents towards local planar structures, the flock alignment ensures that agents also coordinate their movement direction. For each agent i, the alignment force is given by the average velocity (heading) of all its flockmates.

$$F_{align,i} = \frac{1}{N_i}\sum_{j \in N_i} v_j \tag{10}$$

Where:

$N_i = \{j : r < r_{percep}\}$: set of neighbours within the perception distance

v_j : velocity of neighbour j

3.1.5 Total force computation

At each timestep, the net force acting on an agent is computed as the weighted sum of three main behaviour components: eigen-modulated cohesion, eigen-modulated separation and alignment. The eigen-modulated forces bias agent movement along the locally estimated planar directions. Therefore, the total force for each agent is given by:

$$F_{TOTAL,i} = \left(w_{coh} * F_{coh,i}^{eigen}\right) + \left(w_{sep} * F_{sep,i}^{eigen}\right) + \left(w_{align} * F_{align,i}\right) \tag{11}$$

Where:

$F_{coh,i}^{eigen}$: the eigen modulated cohesion force

$F_{sep,i}^{eigen}$: the eigen modulated separation force

$F_{align,i}$: the alignment force

w_{coh}, w_{sep}, w_{align} : scalar weights that influence each behaviour, ranging between 0 and 1. These weights can be adjusted dynamically through the user interface at any time during the simulation.

3.1.6 Movements and rotations

Once the net force F_{TOTAL} is computed for each agent, it is used to define a desired velocity needed to steer the agent towards a new direction. The desired velocity is given by the summation of current velocity and net force.

$$v_i^{desired} = v_i^{current} + \left(\frac{F_{TOTAL,i}}{mass_{agent}}\right)\Delta t \quad (12)$$

The steering is done through quaternion rotations using a SciPy class (SciPy.spatial.transform.Rotation) in python, which ensures smooth and consistent motion in 3D space without gimbal lock issues associated with Euler angles. Mathematically, to perform the rotation, a quaternion is first constructed from the axis-angle representation between the current and desired velocity directions. From this quaternion, a rotation matrix R can be derived using the quaternion identities (de Kemp, 2021).

This rotation is used to apply 3D rotations to both the velocity and normal vectors of each agent. To ensure that these two vectors remain orthogonal throughout the simulation, the same quaternion rotation is applied to both. Consequently, whenever a velocity is updated, the normal vector is rotated at the same time, using the approach described in Figure 3.2.

To ensure a smooth and spherical transition between the current and desired velocity directions, Spherical Linear Interpolation (SLERP), which provides a constant angular directional interpolation, is applied. See appendix B for details.

3.2 Phase II. Spatial estimation with sparse data

In this phase, agents were tested on real geological datasets to interpolate and extrapolate structural information in sparse data environments. Unlike previous exploratory approach in Phase I, where agents were initialized randomly, here each agent was initialized with partial knowledge. Specifically, initial agents were spawned directly from known data points, inheriting the position and directional attributes of their parent data point at the time of spawning.

The primary objective of this phase is to perform structural estimation when data is extremely sparse. To attain such conditions, only a small subset of data points included both the position and normal orientation; referred to as on-contact data, while another subset contains only positional information, referred to as trace positions in this paper.

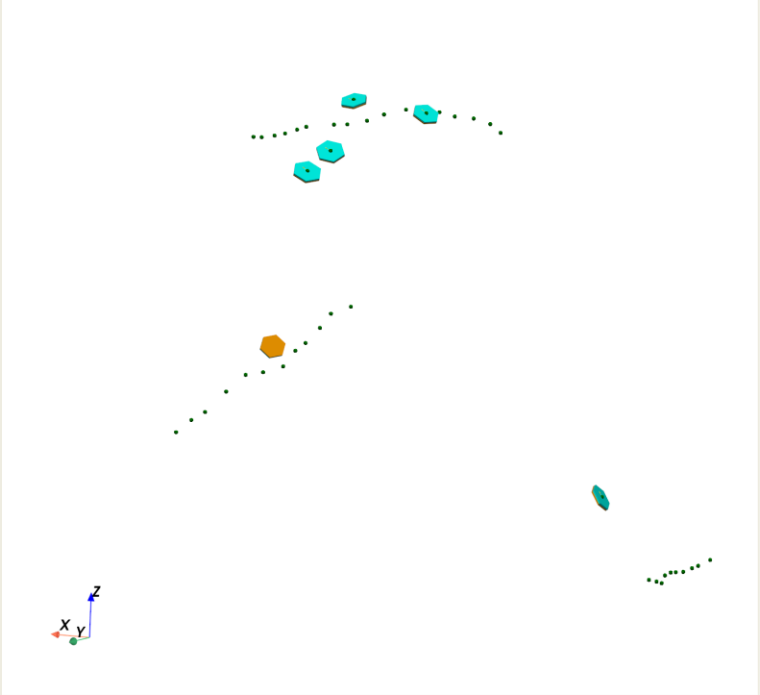

Figure 3.6: Example of sparse data conditions for an overturn structure

As shown in Figure 3.6, on-contact data (with both position and normal orientation) are represented using glyphs: cyan colour for the top section and orange for the bottom. These glyphs are also oriented according to their respective normal directions, the same

way agents' glyphs are oriented. The remaining points in green correspond to trace positions that contain only location data without any associated normal orientation.

3.2.1 Initial normal orientation estimation for trace points

To estimate missing information at trace positions, an inverse distance weighted (IDW) approach was applied. A k-d tree was constructed from on-contact data points to identify the k nearest neighbours to each trace point. If an exact spatial match was found, the corresponding normal was directly assigned, indicating the trace position coincide with an on-contact data. Otherwise, normal orientations from neighbouring points within a specified radius were interpolated using IDW. In cases where no neighbours fell within the radius, the fallback used all k nearest neighbours.

This process, implemented in *process_initial_normals* method of the simulation code, allows to have a starting framework.

3.2.2 Agents' spawning logic

After estimating the missing normal orientation in the dataset using the process described in Section 3.2.1, each data point (on-contact or trace with interpolated normal) is given the ability to spawn N number of agents using radial sampling approach. The data point selects a spawn position on its own plane with a slight offset from its location. If this position respects the minimum distance to other data points, agents and simulation boundaries, it is accepted as a new agent's position. To initialize its normal, a Kent distribution is fitted to neighbouring normal orientations when at least two neighbours are found; otherwise, the parent normal is reused. A candidate orientation is then sampled following "*The 8-parameter Fisher-Bingham distribution on the sphere*" article (Yuan, 2021).

Likewise, each agent is allowed to spawn N number of new agents. However, since the uncertainty increases with each generation, agent-to-agent spawning is performed using the Particle Swarm Optimisation (PSO), which enables to find the best candidate position and orientation while satisfying multiple constraints.

For each spawning agent, a local neighbourhood is defined using nearby stationary data and other agents. The best fit plane is computed using a local PCA, defining a reference

surface. Candidate spawn positions are then optimised using a PSO framework where each particle evaluates the cost of a position based on four components:

- Surface constraint: penalizes deviation from the local best-fit plane,
- Separation constraint: ensures a safe minimum distance from all known data points and agents,
- Radial constraint: penalizes positions too close or too far from the parent,
- Plane consistency: discourages large deviations from the parent's plane.

The surface constraint is assigned the highest weight, followed by the separation constraint. The radial and plane consistency constraints are given equal, lower weights. The candidate position that minimizes the total weighted cost is selected as the new agent's location. Figure 3.7 illustrates the spawning process of an agent (cyan) with 2 neighbours (red). The yellow points represent the PSO particles at their initial positions.

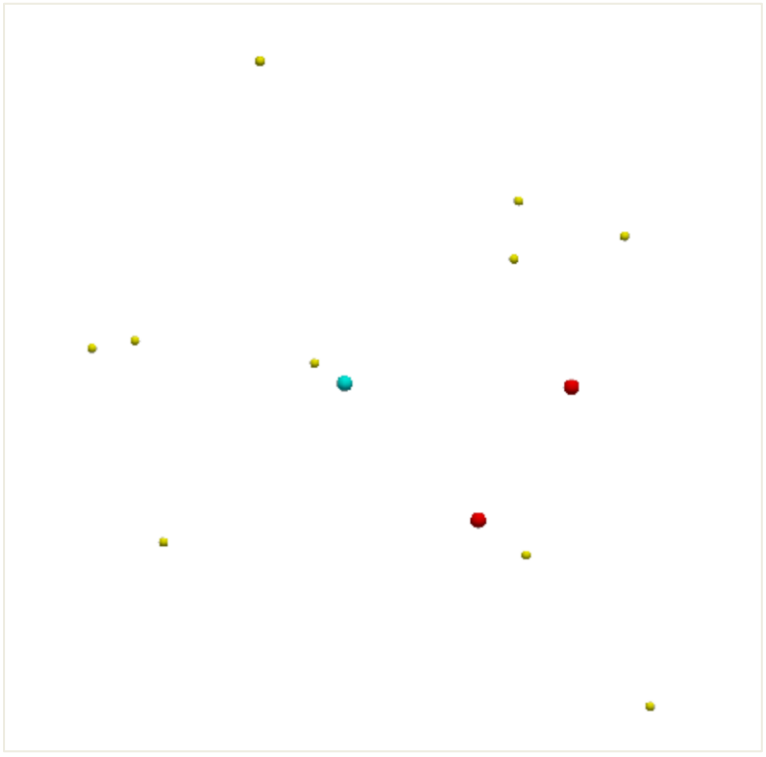

Figure 3.7: Starting conditions of an agent ready to spawn

After the PSO is run until convergence (or the first 100 steps), a spawn position is chosen, that respects both local constraints and the separation distance, as shown by the green point (global best position) in Figure 3.8.

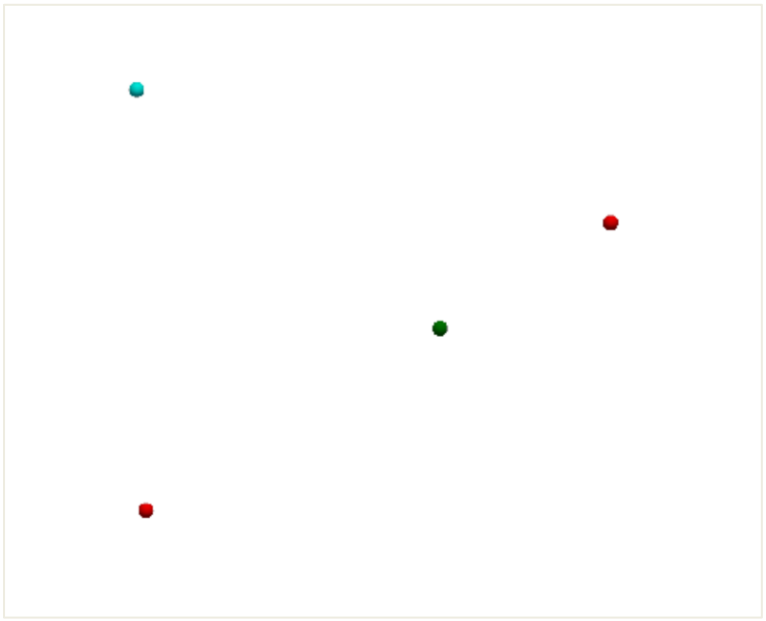

Figure 3.8: Chosen spawn location

Once a valid position is found, the agent's normal orientation is optimised separately using a second PSO. A Kent distribution is fitted to the normal of nearby agents or data. The cost function for orientation minimizes the negative log-likelihood under this kent distribution. This ensures that the new agent's orientation remains locally consistent with nearby structural trends, while optionally allowing constraints on angular deviation from the parent's normal. Finally, the agent's initial velocity is set to be orthogonal to its normal orientation, following the same logic used when initializing normal from velocity directions as described in Phase I (see Figure 3.2).

3.2.3 Neighbour selection

In both the spawning and local-update routines, each agent constructs a combined neighbourhood from on-contact data points, interpolated trace positions, and existing swarm members. A k–d tree is built over this merged set of positions, and every candidate within the perception radius is retrieved. Next, an angular consistency filter is applied: the angle between the focal agent's normal and each candidate's normal is

calculated, and any neighbour whose misalignment exceeds the configured angle_align_tolerance is discarded. This selection by distance, then by orientation ensures that only neighbours exhibiting both spatial proximity and directional coherence contribute to agent spawning and swarm-control updates.

3.2.4 Swarm control

To manage population growth and to ensure structural coherence during surface reconstruction, agents are organized into swarms and updated in local swarm loops. A swarm is defined as a connected component of a graph whose vertices are the current agents and whose edges are agents that meet local consistencies determined by:

i. Spatial proximity: agents are within the perception radius
ii. Directional proximity: angle between agents normal is below a user defined threshold (60^{o} by default).

This construction helps to suppress noise (e.g., diverging agents) and to build consistent local patches in zones with high structural variations (e.g., near faults). The graph is constructed using Networkx (https://networkx.org/), and connected components above a minimum size are retained as swarms. Because positions and orientations evolve over time, the graph (and thus swarm partition) is recomputed as the simulation proceeds, so swarms can split, merge or stabilize as the geometry emerges.

Within each swarm, growth is governed by a simple spawn and prude update implemented in the *Swarm_control* class. The goal is to bias expansion toward locally coherent patches anchored by data, remove agents that contradict the emerging surface, and propagate information smoothly while minimizing oscillations.

Spawning control is restricted to eligible parents, agents that have reached locally defined stopping conditions and have not spawned before. This eligibility gate prevents the same agents from repeatedly seeding new agents allowing a nearly uniform growth within a swarm. The sampling rate is proportional to the swarm size (5% of the swarm is drawn) with a protective floor of at least a small constant to allow some growth at early stages.

To maintain surface consistency, agent pruning remove points that violate local planarity. For each agent, a neighbourhood is taken within the perception radius (or the k nearest

within that radius), and a best-fit plane is estimated by PCA on the neighbourhood positions.

Let $\hat{n}$ denote the plane normal (thirst PCA eigenvector) and $\bar{x}$ the neighbourhood centroid. The orthogonal residual given by:

$$r_i = |(x_i - \bar{x}).\hat{n}| \tag{13}$$

measures the deviation from local surface. Agents whose residual exceeds a threshold (default 3.0 units) are flagged for removal. This step eliminates outliers and sharpens the patches that will be triangulated. The threshold can be relaxed to accommodate gentle curvature or tightened to enforce very flat local neighbourhoods.

Finally, local motion is then applied with reduced velocity for flocking. The same Phase-I behaviours (alignment, cohesion, and separation) act within each swarm, but with a very small step to prevent agents from clustering in biased location at early stages while no reliable surface patches have formed. Neighbours used in these updates already satisfy both distance and angle criteria, so information tends to propagate along consistent directions, where agents anchored to known data (on-contact, or positions that have inferred orientations) influence their neighbours, which in turn steer more distant agents.

3.2.5 Off-contact data

Off-contact data refer to observations collected away from the target surface. These points have both a known spatial location and an associated normal orientation, which can be used to guide agent movement and surface reconstruction. In this study, off-contact data are only used in combination with either on-contact trace locations (without normal orientations) or on-contact data with normal orientations.

When combined with trace locations only, agents are initially spawned from the off-contact points and allowed to propagate until either a single large swarm forms or 100 simulation steps are reached. Following this, agents are attracted toward the trace points with a force scaled by the inverse square of the distance, while maintaining normal cohesion as defined in the SwarmControl class. This approach allows the simulation to first populate the model space with orientation-informed agents before transferring this directional information toward the trace locations.

Chapter 4 Graphical user interface for simulation control

A custom graphical user interface (GUI) was developed to facilitate real-time interaction with the agent-based model. The GUI provides options to launch simulations in two modes: (i) with agents only (as described in Section 3.1), or (ii) with data integration (as outlined in Section 3.2).

If no input data is selected or activated for on-screen display, a warning message is issued, and the simulation proceeds with agent-only mode by calling NoDataABM.py script.

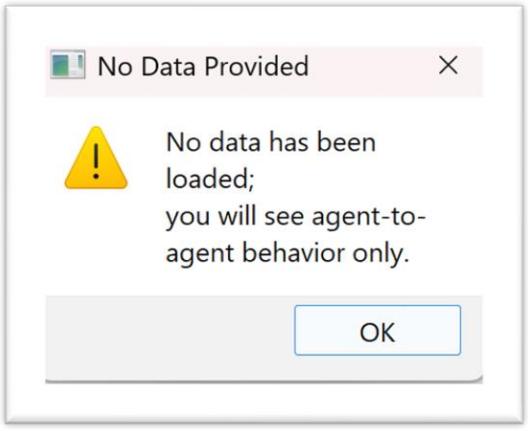

Figure 4.1 Warning message when no data is available

If input data is available, AgentBasedModel.py script is executed, allowing agent-to-data interaction for structural estimation.

4.1.1 Development stack

The graphical user interface was developed with a combination of PyQt5 and PyVista, two Python libraries used in scientific applications. PyQt5 was selected as the primary framework for building the interface components due to its flexibility for stable, cross-platform toolkit with an extensive widgets, layout managers, and event handling. This allowed the integration of sliders, spinboxes, colour pickers, and dialog windows for a user-friendly platform and real time parameters control.

For 3D rendering, PyVista was used as a high-level wrapper around the Visualisation Toolkit (VTK). While VTK is an established industry standard for scientific

visualisation, its low-level C++ interface can be complex. User-friendly platforms such as Paraview (https://www.paraview.org/), also built on top of VTK, are often used for heavy scientific visualisation. However, due to their limited flexibility and integration with custom algorithms, they were not suited for interactive experimentations with agents. PyVista offered the benefits of VTK with a higher flexibility for mesh manipulation, custom glyphs, while easily integrating with PyQt5 using PyVistaQt library through the QtInteractor class. For detailed documentation refer to: https://docs.pyvista.org/index.html.

The interface relies on Pandas and NumPy for efficient data handling. Pandas provides robust tools for importing and parsing csv files, allowing users to map spatial coordinates and normal orientation vectors, by defining default columns or letting users choose appropriate columns. These values are then converted into NumPy arrays, which integrates well with PyVista for visualisation and with the underlying numerical computations.

Figure 4.2 shows the overall architecture of the graphical user interface

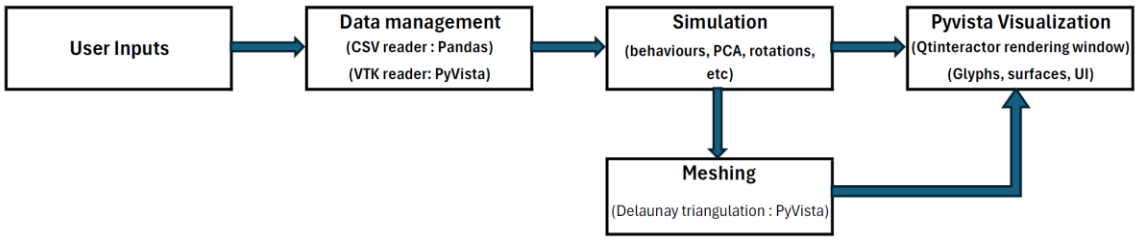

Figure 4.2 GUI architecture

4.1.2 Key features

The user interface was developed in the GUI_Agents.py script and provides interactive control over the simulation. The main features of the GUI are described below, corresponding to the numbered elements in Figure 4.3:

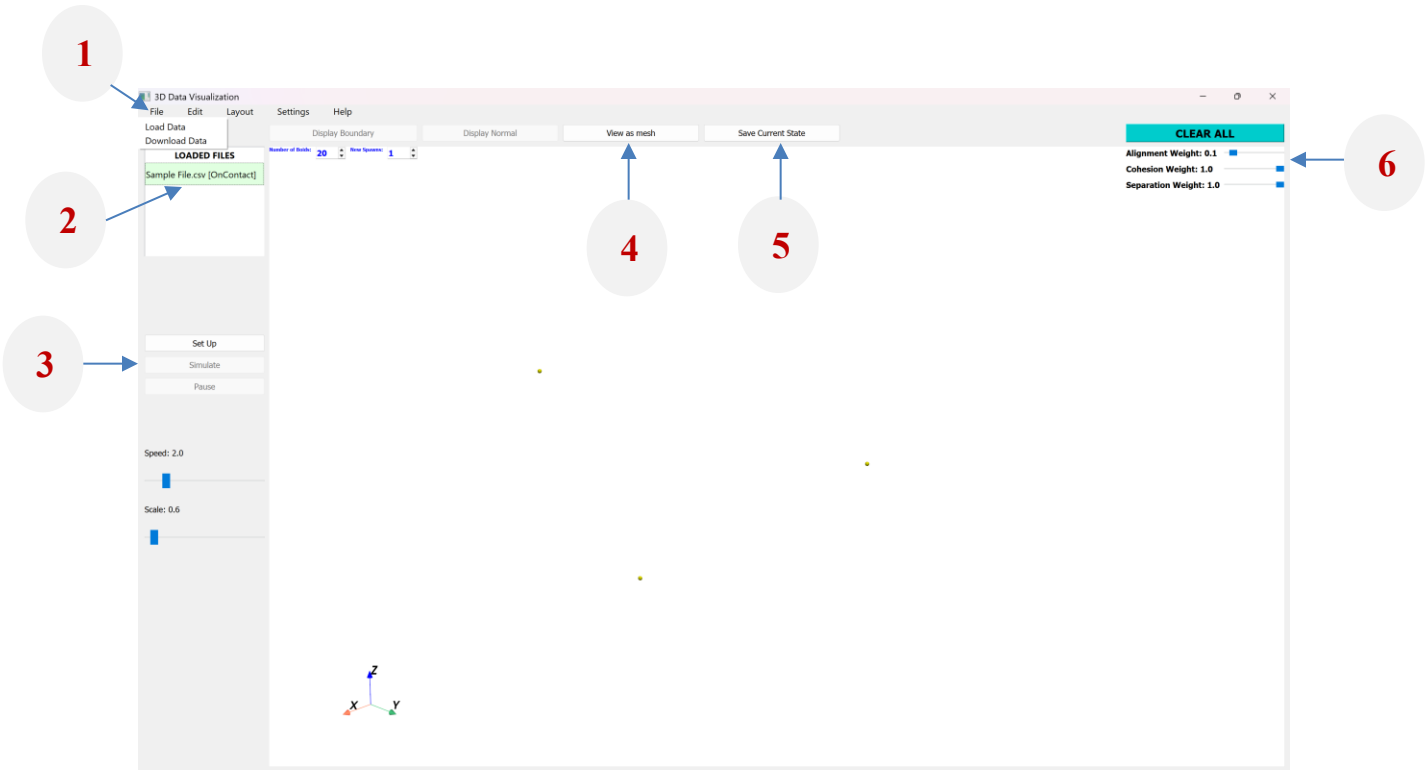

Figure 4.3 GUI key features

1. Menu Bar

Menu bar provides access to files operations, such as loading data sets, downloading data and general settings. The acceptable files for data are csv and vtk files. The GUI offers also an option to load a background mesh for viewing. Upon selecting a file, the user is prompted to specify whether the file is intended for simulation (Data) or for visualisation only (mesh). For simulation input, further distinction is made between On-Contact, Off-Contact and Trace Position.

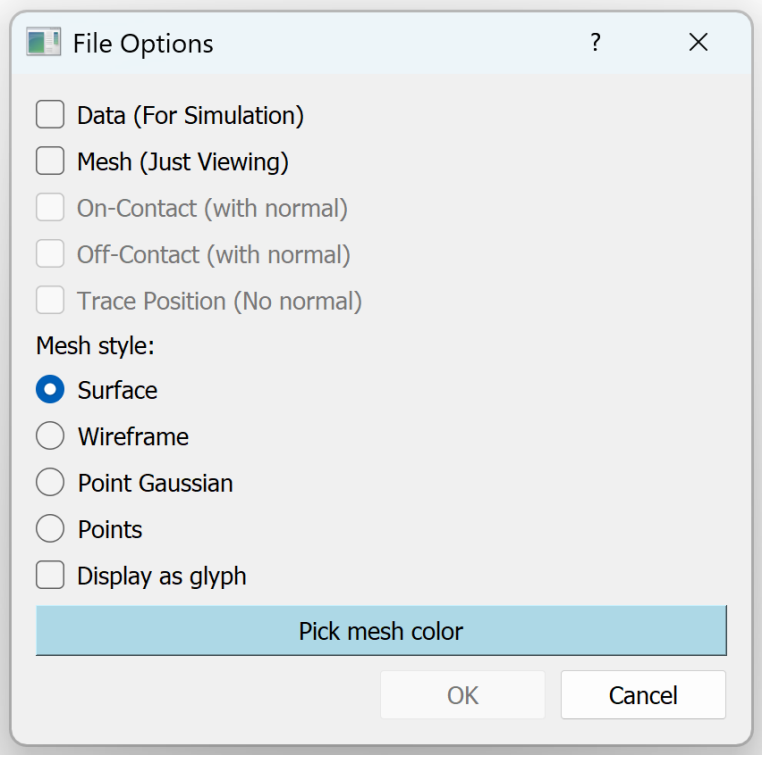

Figure 4.4 File option window

To avoid any confusion and app crashes, the data types are only enabled when the file is selected as data.

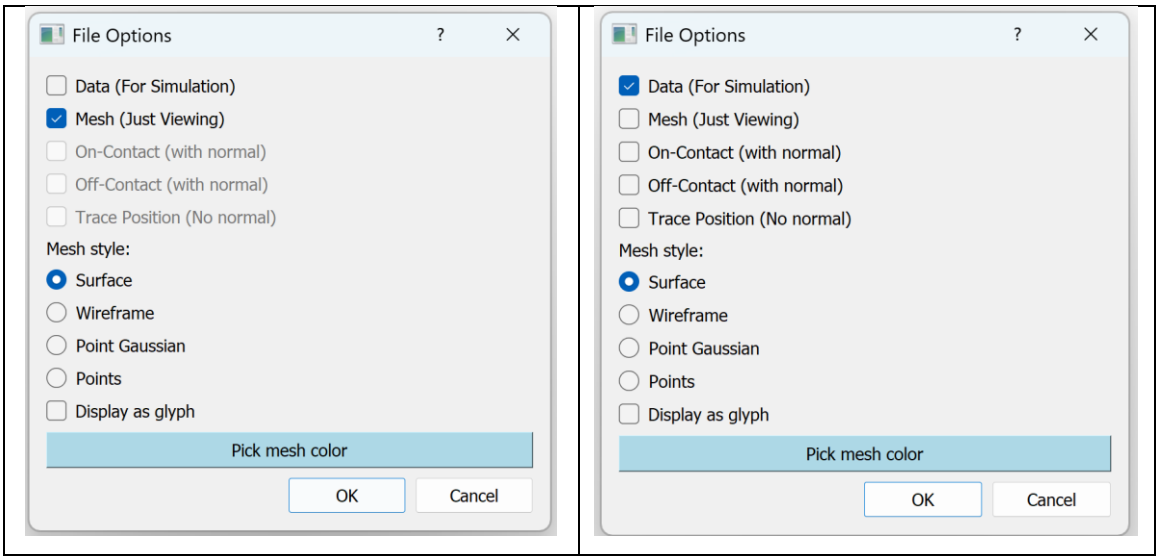

Figure 4.5 Data type selection, where the right image is when mesh is selected and data types are disabled, while on the right, the file selected as data enables data type selection.

When the loaded file is a csv file, the default column names of (X,Y,Z) for positions and (normal(1),normal(2),normal(3)) for normal orientations are proposed by default if they exist in the file with the options to modify. If trace positions, only x,y,z are required, normals can be left as they are.

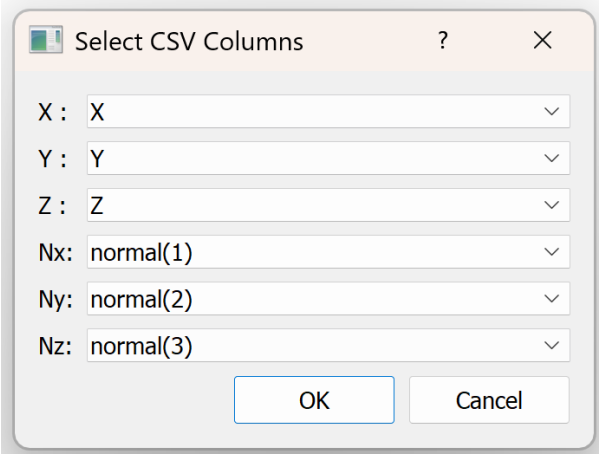

Figure 4.6 csv column options

For vtk or vtp files, the application looks for mesh.points for positions, and mesh.point_data[“normal”] for normal orientations. Similar to csv, orientations are not required if a file is marked as trace positions.

Finally, Mesh style can be specified or adjusted later with a right click on the loaded file name and selecting properties (see loaded files window for details).

2. Loaded files window

As files are loaded to the platform, they are not automatically displayed on the screen. The user must click on a file to make it active file. If a file is marked as data, activating it will make it be considered as input for the simulation. Multiple files can be loaded simultaneously, and files of the same type are concatenated when selected as simulation input.

Here the mesh style (points, surface) as well as the colour can be specified or modified from original selection

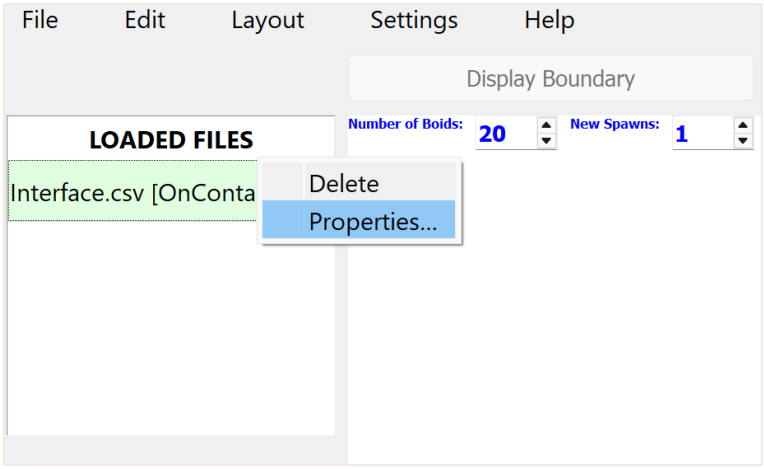

Figure 4.7 Loaded files manipulation

3. Simulation control panel

This panel allows users to configure and manage the simulation. When set up is clicked, the simulation is initialized, allowing to view initial conditions with only data before any interpolation or extrapolation. For agent-to-agent simulation, the Set up button will simply show initial state of agents. At anytime during the simulation, the pause button can be clicked to view the current state.

4. View as mesh button

This control converts agent state (3D positions and normal orientations) into a triangulated surface for inspection and export. The meshing pipeline proceeds in stages. First, agents are clustered by normal orientation using DBSCAN with a cosine metric, which is effective to noise detection and does not require the number of clusters a priori. The angular tolerance θ (in degrees) is mapped to DBSCAN's neighbourhood radius via

$$\varepsilon = 1 - cos(\theta) \tag{14}$$

so that agents whose normal differ by at most θ fall into the same cluster while isolated points are labeled as noise and ignored. For each remaining cluster, VTK's 2D Delaunay triangulation is applied via delaunay_2d. Patches are rendered semi-transparent with

edges to aid interpretation, and basic mesh properties such as face normals and area can be delivered for subsequent analysis.

5. **Save Current State Button**

This button exports the current simulation state, including agent position and normal orientations, to a file. The user can choose between saving as a .csv file containing raw coordinates and normal vectors, or as a .vtp files containing a mesh surface generated in a similar way as the View as mesh button does.

6. **Control sliders**

These sliders provide an interactive way to adjust the behavioural weights and scaling parameters used in the simulation. These include alignment, cohesion, and separation well, as well as speed and glyph scaling. The sliders are most effective when running simulation based on agent-to-agent interactions, where boid movement is governed by primarily by these parameters.

7. **GUI limitations**

In its current form, the interface is aimed at interactive exploration rather than full production meshing. At least one orientational dataset is required to run the data-driven simulation; trace-only points are visualized but cannot drive the controller alone. Therefore, running the model with only trace points will result in an error. Meshing from the View as mesh button reconstructs patches via normal-based clustering and a 2D Delaunay in each cluster, which is effective for inspection but does not yet add a way to mesh globally. In addition, the performance depends on data size and glyph density, where very dense point cloud reduces frame rate, so down sampling and smaller glyphs are recommended.

Because input datasets arrive with heterogeneous and sometimes unknown units (e.g., metres, feet) and scales, the current viewer renders them at native scale, which can make sizes, glyphs, and slider settings look visually inconsistent. In forthcoming updates, an explicit spatial-transform layer will be added. On load, each dataset will be shifted and isotopically scaled into a normalized display frame to stabilize camera behaviour and

visual sizing; a 4×4 affine transform will be stored so all computations run in normalized space, and the exact inverse transform will be applied at export to write results back in the original units. The UI will expose unit selection, a scale bar and axis labels, and options to preserve aspect ratio and enforce a Z-up convention. Perception radius, step sizes, and glyph scales will be interpreted in normalized units with sensible defaults, improving comparability across projects. Normalisation does not alter topology or relative geometry; it only standardizes the visual frame, with a guaranteed back-transformation before saving.

Chapter 5 – Results and discussion

This chapter explores the application of structural agents to reconstruct natural surfaces from sparse data, specifically in geological surface modelling. As discussed in Chapter 2, current approaches, including explicit and implicit modelling, struggle to capture local variability when data is sparse as they tend to rely on global trends. On the other hand, agent-based modelling has shown potential to use a decentralized approach to adapt to local changes while preserving global coherence.

5.1 Preliminary tests without control data

In Phase I of the study, agents' interactions were tested without data integration, assessing the cohesion, separation and alignment of their directional vectors (normal orientations and velocities). The results showed that through local swarm communications, agents were able to stay together (cohesion), maintain a minimum distance from each other (separation), and move in the same direction (alignment). Initially, following the classic Boid algorithm, where an agent tries to head to the centroid of its neighbours, disturbances such as simulation boundary collisions or conflicting neighbour directions would break the cohesion, causing agents to leave their local swarms or to behave abnormally. To improve this, the cohesion was reformulated as a gravitational-like attraction, where each agent exerted an attractive force on its neighbours, scaled inversely with the square of their distance. This ensured that nearby agents influenced each other more strongly than those farther away, creating spatial autocorrelation behaviour within the swarm. Figure 5.1(a) shows the initial state of agents (random starting positions), and Figure 5.1(b) shows agents after they have reached a cohesion state while respecting the separation and alignment rules.

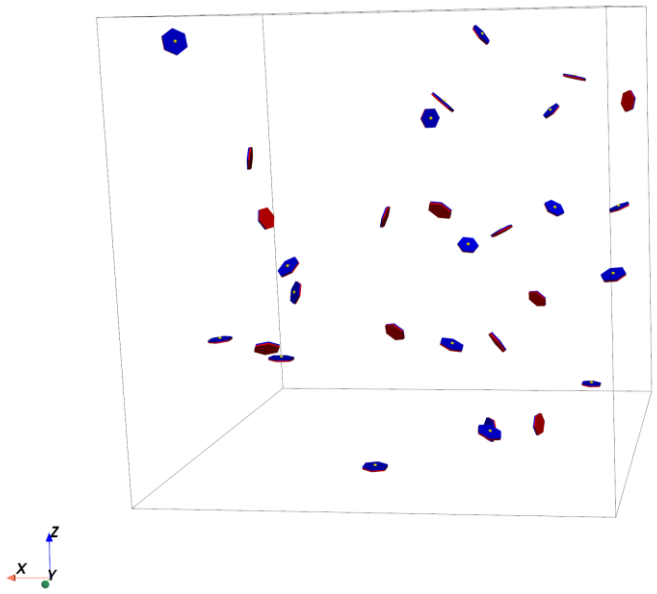

Figure 5.1(a) Initial state of 30 agents initialized at random positions within the simulation boundaries

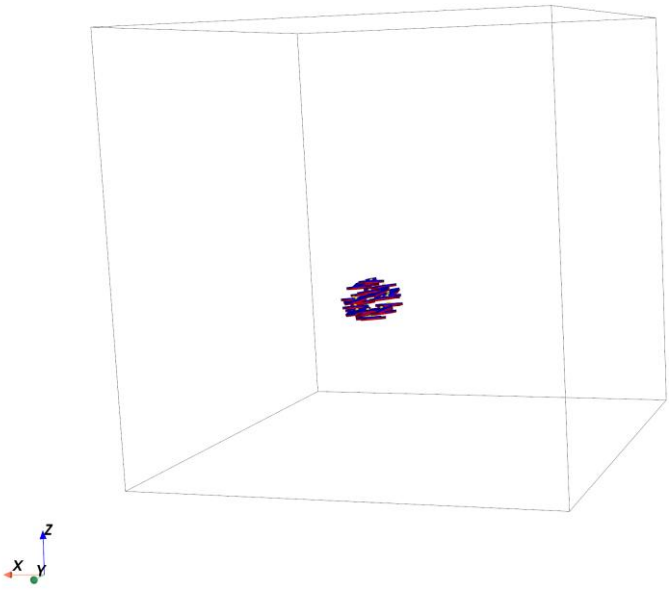

Figure 5.1(b) Agents demonstrating orientation coherence and spatial clustering

Due to the random nature of starting conditions, it takes a different number of steps to reach the desired behaviours, if the simulation is run multiple times. However, for the same behavioural weights (adjustable in the user interface) agents showed similar behaviours after converging into the same swarm. As these agents have no control data, the evaluation of their cohesion was done visually over multiple simulations.

As discussed in Section 3.1.3, through eigen decomposition, agents were spatially biased to create a planar swarm by assigning directional weights. Over time, as agents remained constrained within the simulation boundaries, swarms merged into one plane, composed of independently behaving agents, that can slightly bend. This will later be applied when control data is introduced, to smooth the surfaces and to ensure different swarms communications. Figure 5.2 below shows a plane made by 30 agents, with the green pin showing the current normal orientation of an agent.

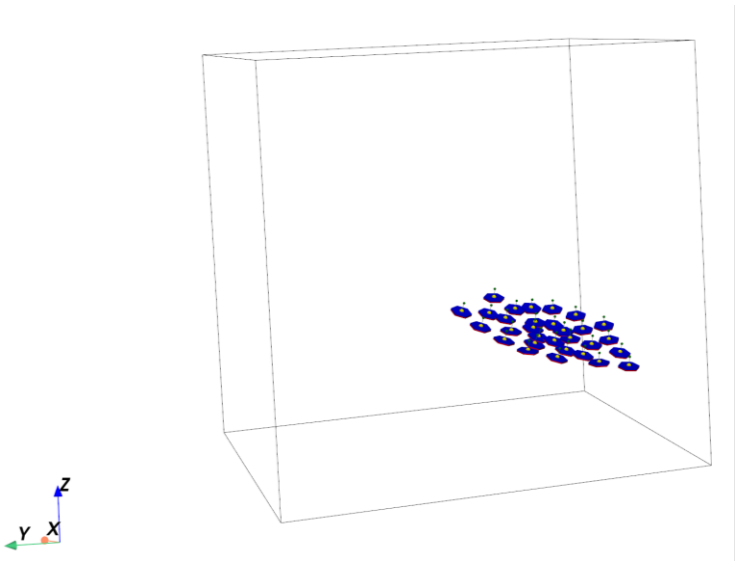

Figure 5.2 Planar structure coherence resulting from spatial biasing

These results demonstrate that agents are capable of self-organizing into coherent structures through decentralized communication, while preserving spatial dependencies. Swarm behaviour was observed to be highly sensitive to the behavioural weights assigned to cohesion, separation, and alignment. Even small adjustments to these parameters had a noticeable impact on the resulting swarm geometry. In addition, the perception radius also played an important role in shaping the structure, where a smaller radius allowed local swarms to bend more flexibly due to limited neighbourhood interaction, but led to increased computational time, as more steps were required to achieve one globally coherent swarm.

5.2 Optimisation of spawning process

In the second phase of this study, control data were introduced, as outlined in Section 3.2, allowing agents to propagate structural information throughout the simulation. Having data spawn initial agents allowed these agents to start with some local information, ensuring directional consistency. When agents were initialized using their spawning parent's or random attributes, such as normal orientation and position, the agent had to independently search for its optimal state before it was eligible to spawn the next agent. As the population grew, this became computationally expensive, often delaying convergence or resulting in non-converging behaviour. This was sensitive due to simulation parameters such as agent speed, or the interpolation factor (t) used in spherical linear interpolation (SLERP), which controls the degree of rotation blending between directions.

Particle swarm optimisation (PSO) was used to estimate both the starting position and the starting normal. A total number of 10 particles were deployed around the ready to spawn agent to perform a spatial search constrained around this agent until convergence (no improvement for 20 consecutive steps) or 100 iterations per simulation step. Through multiple simulation runs, agents appeared to converge in less than 20 PSO iterations per one simulation step, therefore 20 consecutive PSO iterations were considered appropriate to define convergence. Because of the early stopping condition, keeping the maximum PSO iterations to 100 did not add any significant computational complexity but allowed flexibility for more exploration in complex neighbourhoods. Figure 5.3 below shows an example of the cost recorded in a given timestep for 5 random agents to illustrate the chosen PSO patience for convergence.

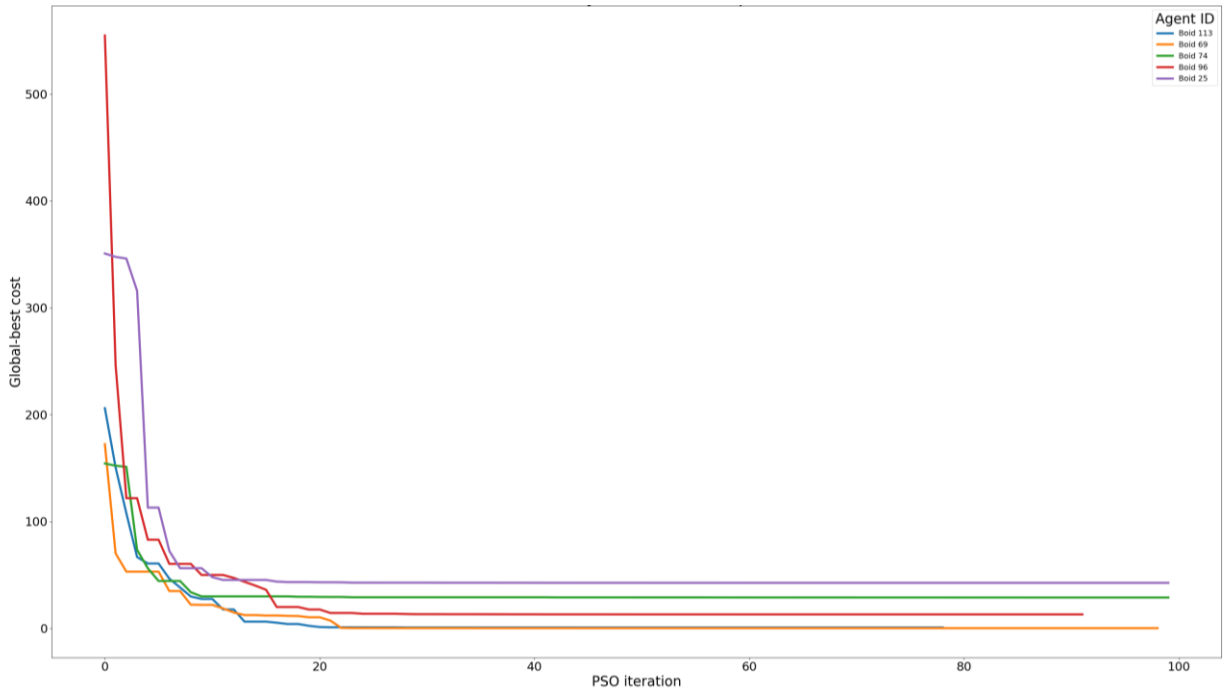

Figure 5.3 Example of PSO results in one simulation step

Using Kent distribution to define each spawned agent's normal allowed to directly model the anisotropic, spherical dispersion of its neighbour's orientations, rather than treating them as if they lived in a Euclidean tangent-plane. However, simply drawing a random sample from the fitted Kent distribution injected stochastic noise into each agent's orientation. This is fine if the model is allowed to run for longer periods of time as the swarm control class will readjust the orientations, but applying PSO that minimizes the negative log-likelihood of the Kent density yielded similar results in a shorter time. This is due to how particles in PSO search for the best-fit normal that respects both the distribution and the local cohesion of nearby agents and data. Therefore, the swarm control class required fewer steps to align agents locally.

5.3 Swarm Scenarios and experimental setup

In geological surface modelling data type is often on-contact or off-contact, and might have corresponding normal orientations or not. Table 5.1 summarises these data types. On-contact data are collected directly on the surface to be modelled and can included a normal orientation or simply a spatial position (referred to as on-contact and trace position, respectively).

Table 5.1 Common geological data types

Data Type	On-Contact	Normal
1	YES	YES
2	YES	NO
3	NO	YES

Figure 5.4 illustrates the types of data points used in this study.

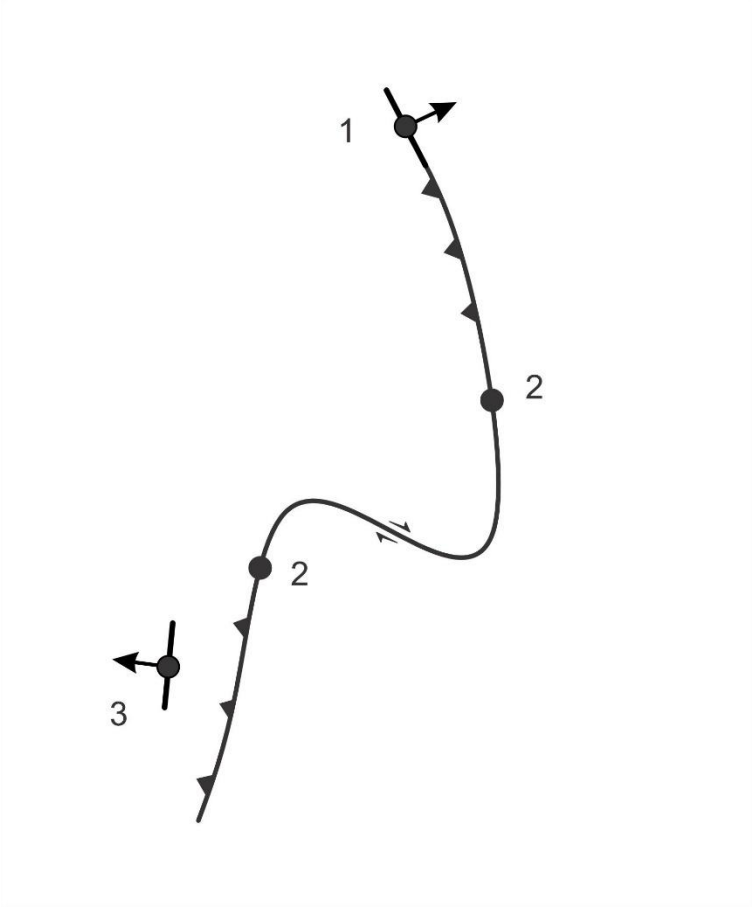

Figure 5.4 Example of different data types (1= on-contact with dip, 2 = on-contact no dip, 3 = off-contact with dip). Curved line represents surface trace of geological thrust fault with dextral strike-slip component, teeth ornamentation on hanging wall.

These swarm scenarios were tested independently and in combination to assess the agent's ability to model surfaces in such sparse data settings. Table 5.2 summarizes the experiments conducted in this study, data type combination, and corresponding case studies.

Table 5.2 Summary of Scenarios

Scenarios	Data Type	Studied	Case Study
Scenario A	1	YES	1st (Synthetic)
Scenario B	2	NO	N/A
Scenario C	3	NO	N/A
Scenario D	1 and 2	YES	1st (Synthetic)
Scenario E	2 and 3	YES	2nd (Docking)
Scenario F	1 and 3	NO	N/A

5.4 Case study 1: Overturn structure from synthetic data

A synthetic dataset representing the extents of a typical regional-scale overturn structure was used to test the method. When evaluated using only the spawning logic, where agents were spawned at selected locations with their best normal estimate determined via PSO and remained stationary, the resulting surfaces showed strong local connection but many swarm patches that failed to merge. This was primarily due to the inability of stationary agents to update their initial estimates as new information emerged. The integration of swarm control class helped solve this, by applying the flocking rules, through a spherical linear interpolation (SLERP), as the swarm grew.

Figure 5.5 shows the sampled structure, with on-contact samples, where the yellow points are trace locations (without normal) and cyan glyphs with red points representing the on-contact with normal. The gray and red colours show the top and bottom faces of the surface, indicating that the structure is overturned. The glyph polarity design (cyan for top, magenta for bottom) further confirms the overturned geometry.

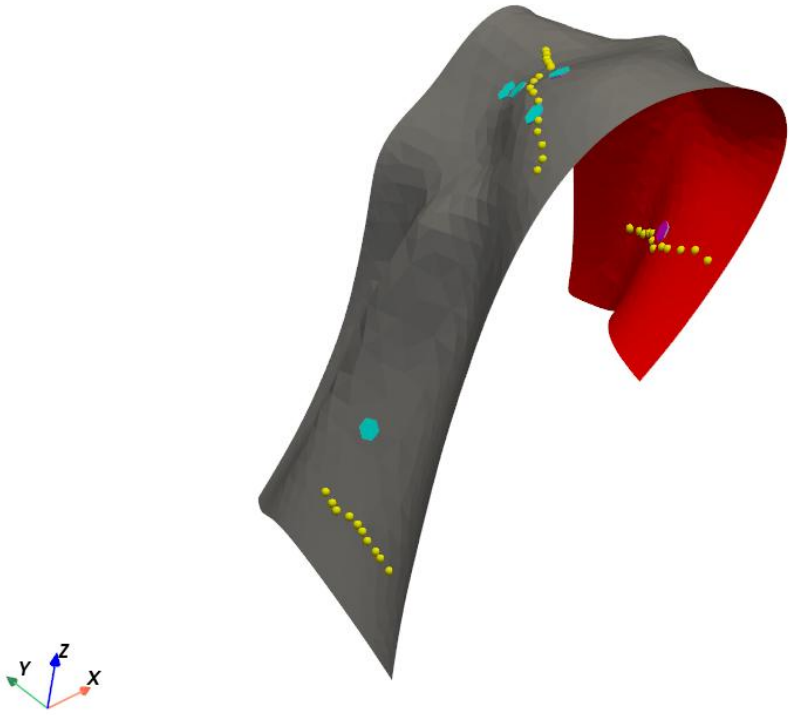

Figure 5.5 Synthetic overturned geologic horizon with constraint data (yellow on-contact traces and local on-contact dips, cyan up-right and magenta overturned).

a. Scenario A: On-contact with normal

When tested on the on-contact data with normal orientations, agents were able to emerge into 2 swarms but failed to connect at the hinge location. In addition, the surface complexity played an important role in how smoothness of the structure was, as shown by the generated mesh in Figure 5.6(b). Figure 5.6 (a) shows starting conditions, before agents are initialized.

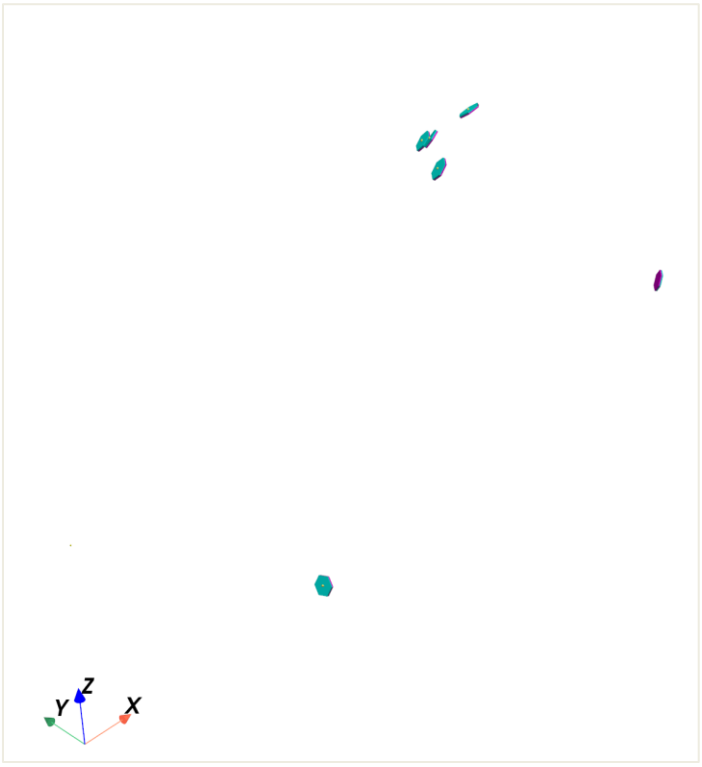

Figure 5.6 (a) Sampled on-contact data

After running the simulation, the resulting structure is shown in Figure 5.6 (b), where the left side displays agents in swarms and the right side shows the resulting mesh.

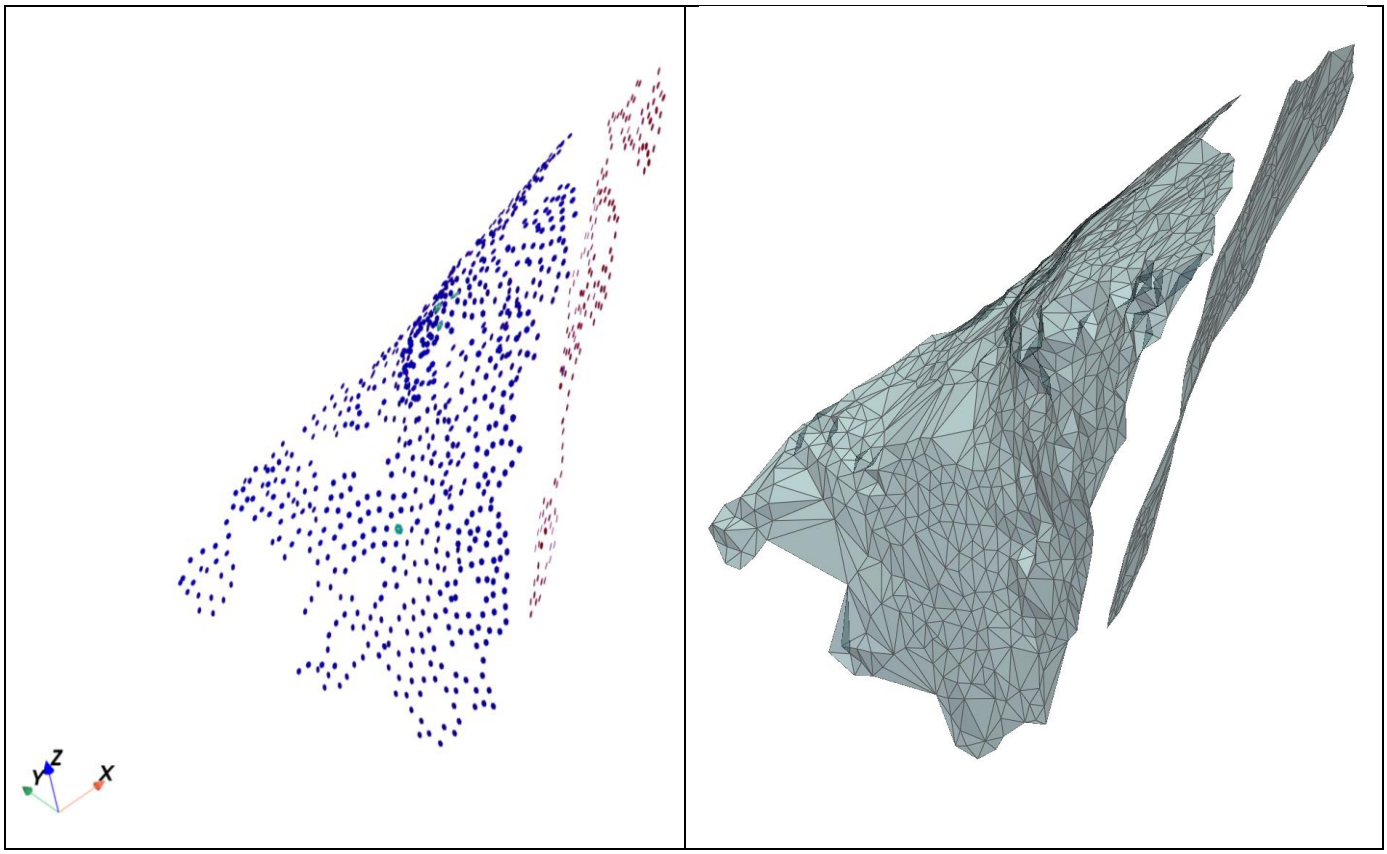

Figure 5.6 (b) Swarm formation and resulting mesh from on-contact data. Notice two independent swarms without global surface cohesion.

This lack of cross-swarm connection is due to the constraint imposed on agents on who to consider as a neighbour, which is: agents and data within perception radius and whose normal orientation is less than the default 90° from the current agent. When this angular constraint was relaxed to let agent consider multiple neighbours as long as they are within the Euclidean perception radius, the resulting structure was not coherent, and the swarm behaviour was unpredictable as agents had a very limited guidance.

b. Scenario C: On-contact with normal and on-contact without normal

When trace locations were introduced alongside on-contact points with normal, the model produced improved results. Although these trace data lacked orientation, they still influenced the overall structure geometry by providing anchors that helped guide the agents. Unlike in Scenario A, where agent swarms trended to connect both around the hinge (top) and along the sides, the addition of trace points appeared to constrain swarm

growth to a more realistic geometry. The initial conditions before agents are initialized are shown In Figure 5.7 (a).

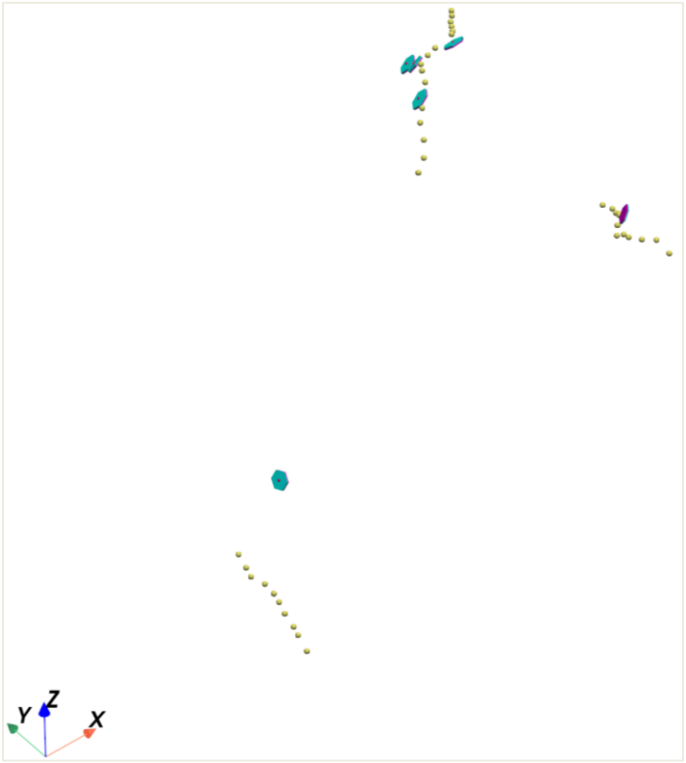

Figure 5.7 (a) On-contact data with normal and trace locations in yellow points

The resulting swarm structure is represented in Figure 5.7 (b), where the right side of the figure shows the swarm of agents and the right side represents the generated mesh.

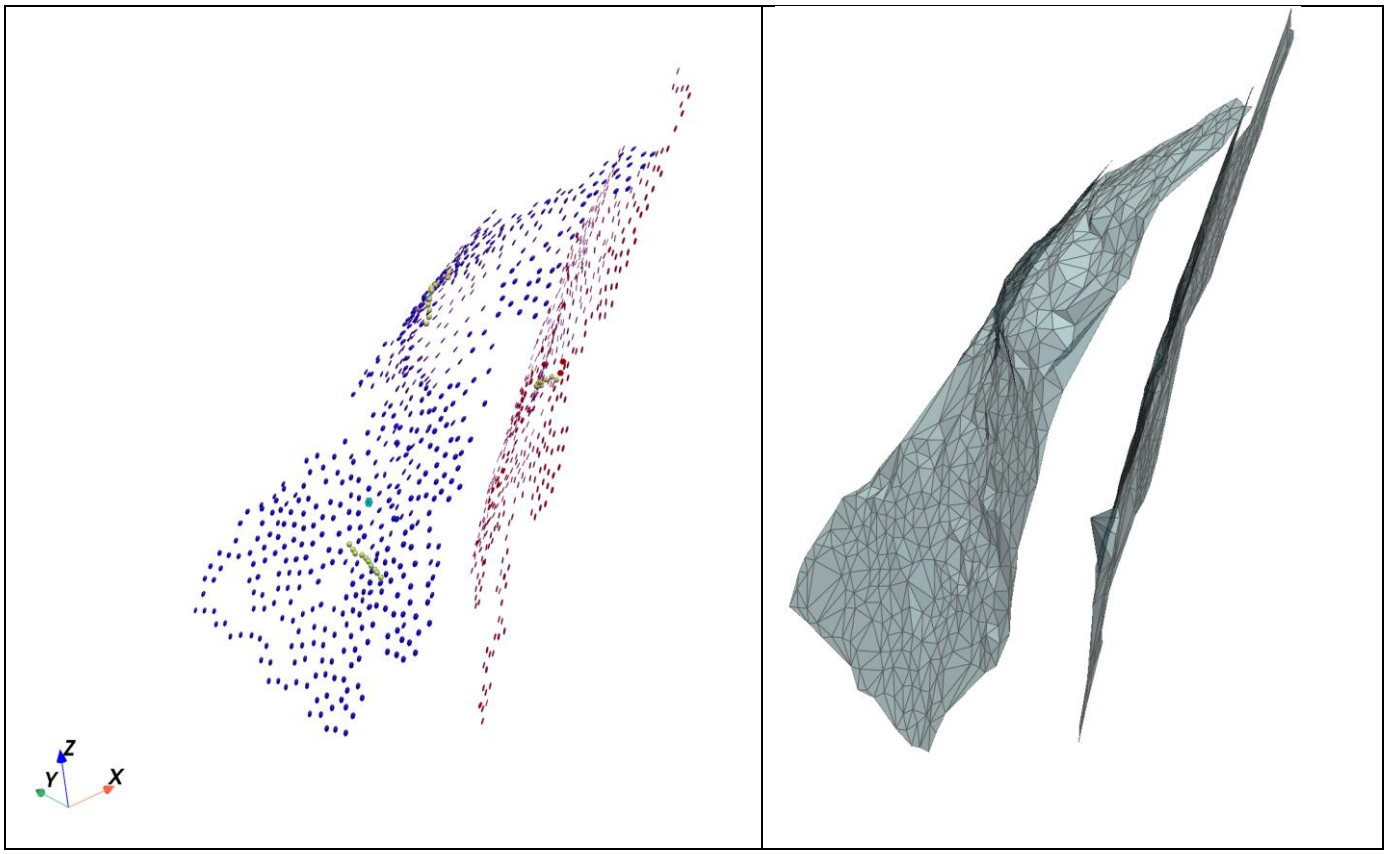

Figure 5.7 (b) Swarm formation and resulting mesh from on-contact and trace data

Scenario A showed the ability of agents to grow from only very limited on-contact information. While not ideal for estimating complex surfaces, in sparse data setting this approach can provide an initial tool to generate synthetic data for more robust models. In contrast, Scenario C showed a smoother and more consistent surface. However, due to the imposed angular constraints in determining a neighbour, both scenarios struggled to smoothly transition at the hinge (where the 2 swarms connect). In Scenario C, removing the constraint while picking the neighbours to compute a spawn normal resulted in agents being able to have a cross-swarm communication but less stable condition and at a cost of losing the surface smoothness or the realism, as the connection can be at undesired locations, unlike what they achieved with the constraint, as shown in Figure 5.8. In this scenario, the chosen perception radius dictates the resulting swarm geometry.

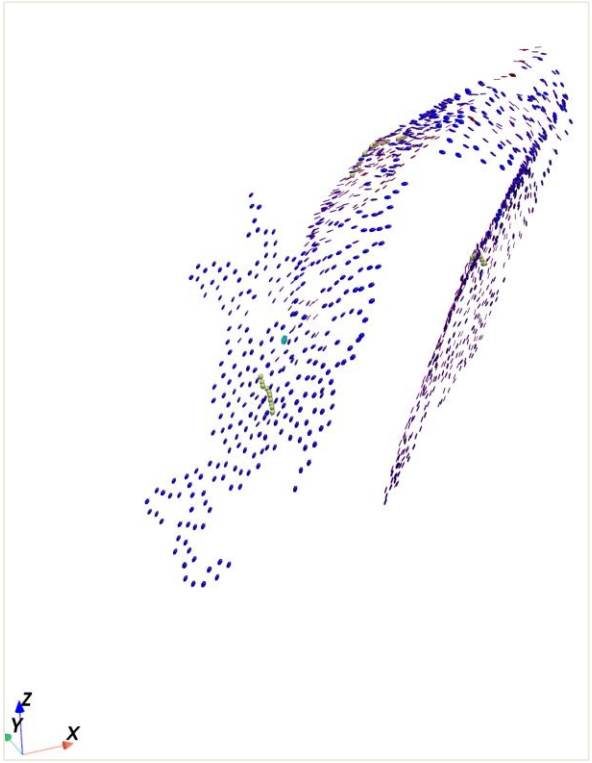

Figure 5.8 Swarm structure without angular constraints

Despite struggling in highly complex regions such as at the hinge location, this structural agent-based model approach, still in its early stage, shows improved results in capturing local variability compared to the implicit modelling approach for this specific scenario, where the surface is overly smoothed but maintains the overall shape, as presented in figure 5.9. Figure 5.9 represents the model out put from SURFE, a geological interpolation tool that uses implicit surface modelling based on radial basis functions to estimate surfaces (Hillier et al., 2014), where the same trace and on-contact data (with normal) were provided as the model input. The complete guide on how to install and run SURFE is available at: https://github.com/MichaelHillier/surfe.

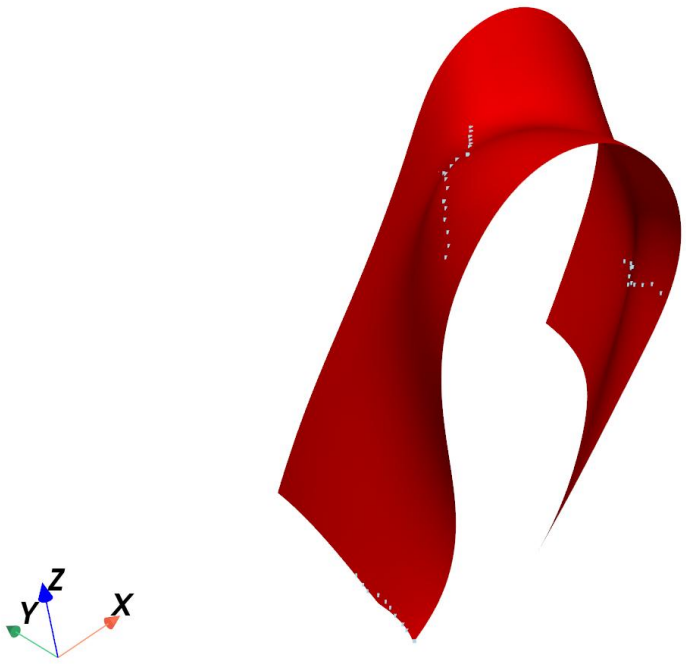

Figure 5.9 Output of SURFE model using the synthetic dataset generated using Thin Plate Spline as the RBF kernel with polynomial order of 1.

5.5 Case study 2: (Scenario E) Docking Formation

This case study applies the agent-based model to the Docking Formation in the Kananaskis region of the Canadian Rockies (Alberta). This is geologically complex area when compared to a similar corelated sequence of rocks in the Western Canada Sedimentary Basin (WCSB). The Kananaskis region is part of the western Canadian Cordillera mountain belt and has undergone intense orogenic deformation producing fold and fault patterns in the subsurface (McMechan, 2013; Thapa & McMechan, 2019). It was previously modelled using the SPARSE package in Gocad/SKUA a suite of explicit and implicit 3D geological modelling tools. The SPARSE package applies an explicit parametric approach to interpolate geological surfaces from grip frame inputs (Sprague & de Kemp, 2005). The reference model covers a 3D model space of 30 km × 20 km × 5 km and serves as a benchmark for evaluating the performance of the current model in real geological settings, where the goal is to have agents estimate the red surface using the

trace locations (yellow points in the figure) and a few off-contact data with a normal orientation. Figure 5.10 shows the explicit model output used as a reference.

.

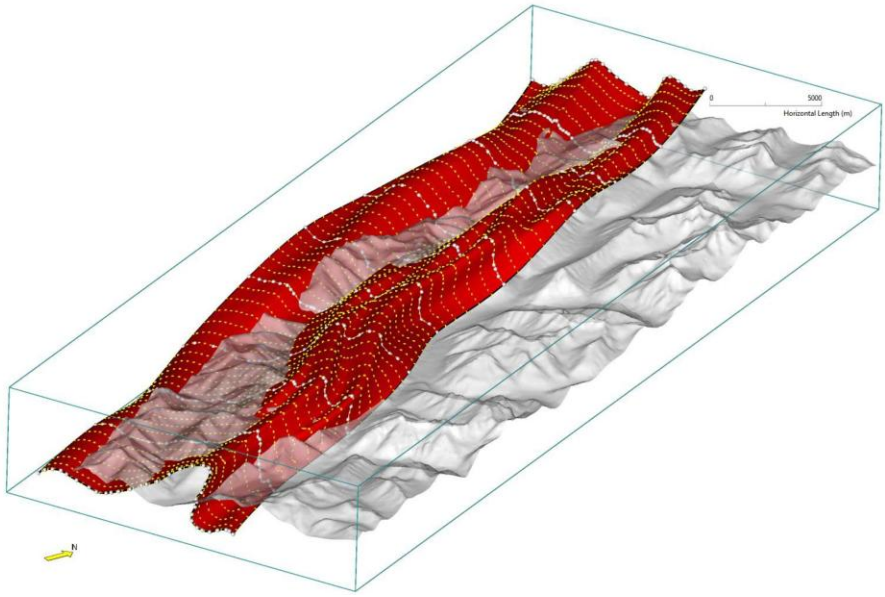

Figure 5.10 Docking Formation modelled with SPARSE

Using available trace data and limited orientation data, as shown in Figure 5.11 for starting conditions, agents were successful in creating one big swarm connecting all the orientation data (off contact). This was the expected behaviour, as agents were instructed to first propagate information from known orientations and migrate to trace locations, thus having an initial spatial guide

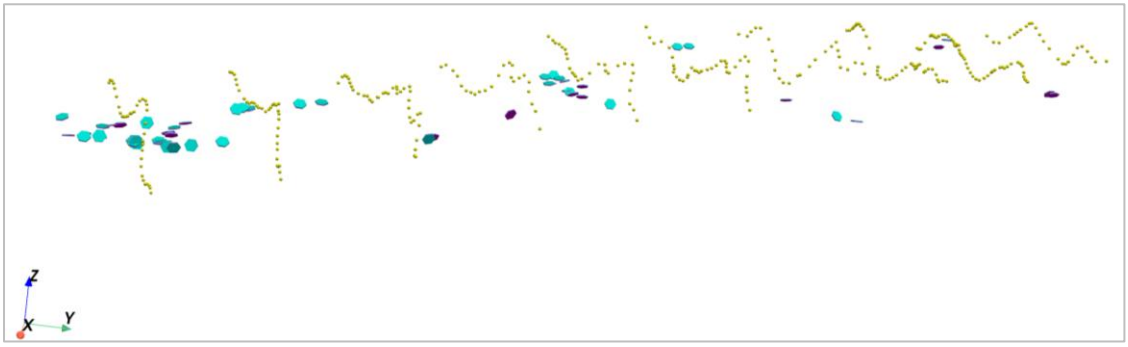

Figure 5.11 Initial states in Scenario E

The resulting swarm formation, generated from off-contact data with normal orientations, before agents start migrating to trace locations, is illustrated in Figure 5.12.

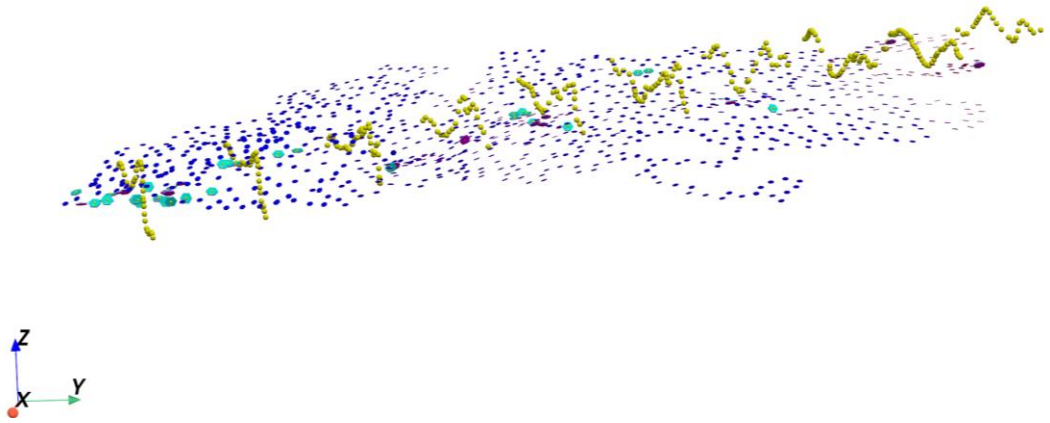

Figure 5.12 Swarm formation from known orientational information

However, as the agents migrated to the on-contact trace location, there was a lack of overall guidance about the normal at those locations, and agents failed to reconstruct a realistically geological structure. Figure 5.13 shows the results of mixing the two data types: off-contact with normal and on-contact without normal.

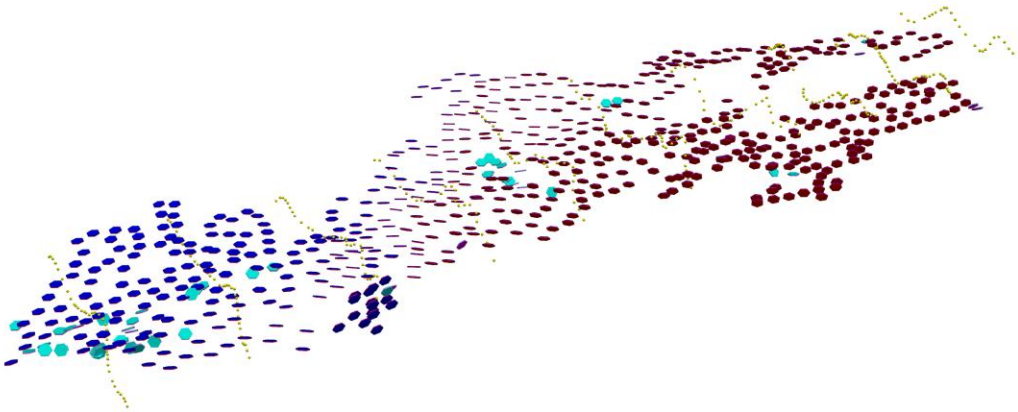

Figure 5.13 simulation results for on contact trace data without normal and off-contact data with normal orientations. Notice the lack of coherent top information, which should all be compatible with an upright fold structure.

Based on the visual comparison with the previous model on the same structure in Figure 4.8, the parametric (explicit) model outperformed the current ABM model. This is due to the lack of incorporating knowledge about specific structures, as the current model has been generalized without clear information about what the end structure should look like, unlike the explicit model which relies on a user-defined grip frame which has its knot sequence organized in a way that preserves the geologic topology.

5.6 Scenarios comparison

Case studies scenarios are summarized in Table 5.3 below, along with the observed issues for each of the scenarios. In addition, based on the reviewed literatures and the observed behaviours of agents, potential issues for Scenarios B and C are presented in the table to be addressed in future work.

Table 5.3 Summary of issues for each scenario

Scenarios	Data Type	Studied	Case Study	Issues
Scenario A	1	YES	1st (Synthetic)	Results with limited success, complexity of surface is a big factor
Scenario B	2	NO	N/A	Implicit and explicit modelling methods, Data density dependant.
Scenario C	3	NO	N/A	Use Implicit methods for vector field calculation
Scenario D	1 and 2	YES	1st (Synthetic)	Works well for local swarms, connection at hinge location is unstable and at a cost of surface smoothness.
Scenario E	2 and 3	YES	2nd (Docking)	Parametric (Explicit) model works better. Swarm patches don't converge as agents migrate to trace locations (without orientation).
Scenario F	1 and 3	YES	N/A	Needs more testing. Agents require different weightings for off-contact data with normal orientations.

As shown in Table 5.3, agents demonstrated strong local behaviour when the available orientational information was sampled directly on the surface of interest. Scenarios that combined on-contact traces with on-contact data containing normal orientations produced

the most coherent results based on visual inspection. In contrast, scenarios driven by off-contact constraints, even when accompanied by orientational information, required significantly stronger cross-swarm communication for agents to migrate toward trace locations without losing cohesion both locally and across the broader structure. This is largely due to the increased uncertainty these data types provide, especially since the model was developed as a generalised framework without prior knowledge of the expected structural geometry. Because geological surfaces can be highly complex, introducing a mechanism that allows agents to recognise or infer the underlying structure type would help reduce uncertainty during migration.

Even though scenarios relying purely on off-contact data and scenarios in which no orientational information is provided were not studied in this research, the observed agent behaviours suggest that such configurations would pose even greater challenges. For example, in scenario B, where only on-contact traces are available, implicit or explicit modelling approaches would likely perform better, provided that the data density is sufficient, as discussed in Chapter 2. For Scenario C, which relies on off-contact constraints but includes orientational information, it would be advantageous to first define a vector field using implicit modelling approaches before allowing the structural agents to estimate the surface of interest. This would provide the agents with the necessary structural orientation attributes, which are essential for their best performance under the current framework.

In addition to these challenges, validation in sparse-data settings presents another important limitation. Sparse inputs provide too few independent observations to rigorously assess accuracy, and as a result, the current framework relied primarily on visual inspection and geological expert interpretation rather than quantitative validation.

Chapter 6 – Conclusions and recommendations

Geological surface modelling remains challenging in sparse and incomplete data settings, where traditional interpolation and modelling approaches often fail to capture local variability or respect geological constraints. This study presented a structural agent-based modelling framework for geological surface reconstruction in sparse data environments. The main objective was to explore the potential of extending agent-based approaches to geological surface modelling and to evaluate their strengths and limitations for long-term applications. By applying decentralized swarm communication rules, agents demonstrated the capacity to self-organize into coherent structures while adapting to varying spatial constraints.

The framework combined the modified local flocking rules of cohesion, separation and alignment with statistical constraints such as PCA for spatial biasing and kent distribution sampling to guide agent orientation. A PSO based spawning process was introduced to select candidate positions for new agents efficiently, allowing many agents in the simulation to interact.

When tested on different scenarios, the model incorporated multiple geological data types such as on-contact with normal orientation, off-contact with normal orientation, and on-contact without normal orientation (referred to as trace locations in the paper). In synthetic datasets combining on-contact normals with trace locations, the model captured local variation effectively but struggled to produce smooth transitions across hinge locations compared to an implicit model. In scenarios using off-contact data and trace locations, the model showed initial success in local connectivity but lacked broader structural guidance as agents migrated toward trace locations due to the lack of knowledge of the final structure type. In such cases, explicit modelling approaches tend to perform better, as they rely on manual configurations and expert interpretations that impose global constraints.

This exploratory research demonstrates that an agent-based approach can complement existing modelling methods, particularly in situations where sparse data and local variability challenge traditional methods. Across all case studies, the structural agent-

based model demonstrated the ability of agents to reconstruct locally coherent surfaces from sparse geological data. The decentralized nature of agent communications allowed the swarm to preserve local and more realistic structures even in absence of dense data input. The evaluation of each data scenario helped to visually observe how different types of geological data influence the emergent behaviour of the agents.

Despite challenges encountered across scenarios, this exploratory study highlights several advantages of structural agent-based modelling in geological surface reconstruction. Agents can not only interpolate known data but also extrapolate surface information within the defined 3D space while maintaining local cohesion. This may increase the accuracy of more traditional tools such as implicit approaches due to the increased data density provided by swarm technology. Additionally, this approach allows for further constraints to be imposed on agent behaviours, and agents can be extended to learn and adapt as the simulation evolves.

However, one persistent limitation was the lack of mechanism for establishing global structural continuity, especially across complex features such as fold transitions. In all tested scenarios, agents relied on local constraints to define neighbourhoods, which limited their ability to resolve highly changing regions. Relaxing some constraints encouraged cross-swarm merging but degraded the surface smoothness, suggesting conflicts between local alignment and global integration.

Future work could explore the integration of new control classes for bridging agents to help resolve the sensitivity observed around complex zone with high inter region structural variations. Moreover, the current system's dependence on fixed global parameters, such as perception distance and angular thresholds, makes it less adaptive in heterogeneous settings. Agents could be enhanced to intelligently adjust their interaction rules based on metrics such as curvature, uncertainty, or neighbourhood density. Incorporating reinforcement learning strategies may also allow agents to refine their behaviour through experience, learning movement strategies that balance cohesion, smoothness and geological realism across varying contexts. In addition, since the present evaluation relied primarily on visual assessment, future development should incorporate quantitative statistical metrics to provide more rigorous performance evaluation. Finally, future research could explore hybrid strategies that couple agent-based modelling with

established implicit methods and emerging neural implicit representations such as GeoINR, allowing improved global structural coherence while preserving the flexibility of local agent dynamics and capturing local variability.

Ultimately, this research demonstrates that structural agent-based modelling can serve as a flexible complement to implicit and geostatistical methods, offering potential pathways for subsurface modelling in contexts such as critical mineral exploration, and geohazard mitigation where data are often sparse, but decisions are at high-stakes.

References

AbdelRahman, M. A. E., Zakarya, Y. M., Metwaly, M. M., & Koubouris, G. (2021). Deciphering Soil Spatial Variability through Geostatistics and Interpolation Techniques. *Sustainability*, *13*(1), 194. https://doi.org/10.3390/su13010194

Alem, Y. F., Khalid, Z., & Kennedy, R. A. (2015). *Spherical Harmonic Expansion of Fisher-Bingham Distribution and 3D Spatial Fading Correlation for Multiple-Antenna Systems* (No. arXiv:1501.04395). arXiv. https://doi.org/10.48550/arXiv.1501.04395

American Geosciences Institute. (2025). *Economic analysis of the costs and benefits of geological mapping in the United States of America from 1994 to 2019*. American Geosciences Institute.

Anand, A., Singh, P., Srivastava, P. K., & Gupta, M. (2021). Chapter 19—GIS-based analysis for soil moisture estimation via kriging with external drift. In P. K. Srivastava, M. Gupta, G. Tsakiris, & N. W. Quinn (Eds.), *Agricultural Water Management* (pp. 391–408). Academic Press. https://doi.org/10.1016/B978-0-12-812362-1.00019-9

Anoune, K., Bouya, M., Astito, A., & Abdellah, A. B. (2018). Sizing methods and optimization techniques for PV-wind based hybrid renewable energy system: A review. *Renewable and Sustainable Energy Reviews*, *93*, 652–673. https://doi.org/10.1016/j.rser.2018.05.032

Auflič, M. J., Herrera, G., Mateos, R. M., Poyiadji, E., Quental, L., Severine, B., Peternel, T., Podolszki, L., Calcaterra, S., Kociu, A., Warmuz, B., Jelének, J., Hadjicharalambous, K., Becher, G. P., Dashwood, C., Ondrus, P., Minkevičius,

V., Todorović, S., Møller, J. J., & Marturia, J. (2023). Landslide monitoring techniques in the Geological Surveys of Europe. *Landslides*, *20*(5), 951–965. https://doi.org/10.1007/s10346-022-02007-1

Barros, J. (2012). Exploring Urban Dynamics in Latin American Cities Using an Agent-Based Simulation Approach. In A. J. Heppenstall, A. T. Crooks, L. M. See, & M. Batty (Eds.), *Agent-Based Models of Geographical Systems* (pp. 571–589). Springer Netherlands. https://doi.org/10.1007/978-90-481-8927-4_28

Barudžija, U., Ivšinović, J., & Malvić, T. (2024). Selection of the Value of the Power Distance Exponent for Mapping with the Inverse Distance Weighting Method—Application in Subsurface Porosity Mapping, Northern Croatia Neogene. *Geosciences*, *14*(6), 155. https://doi.org/10.3390/geosciences14060155

Beni, T., Borselli, D., Bonechi, L., Brocchini, D., Guideri, S., Gonzi, S., Carlà, T., Gigli, G., Lombardi, L., Nocentini, M., Ciulli, V., D'Alessandro, R., & Casagli, N. (2025). Use of muon imaging for enhanced reconstruction of underground cavities and associated stability conditions at a historical mining site. *Engineering Geology*, *353*, 108141. https://doi.org/10.1016/j.enggeo.2025.108141

Bernardes, T., Gontijo, I., Andrade, H., Vieira, T. G. C., & Alves, H. M. R. (2006). Digital Terrain Models Derived from SRTM Data and Kriging. In A. Abdul-Rahman, S. Zlatanova, & V. Coors (Eds.), *Innovations in 3D Geo Information Systems* (pp. 673–682). Springer. https://doi.org/10.1007/978-3-540-36998-1_51

Bingham, C. (1974). An Antipodally Symmetric Distribution on the Sphere. *The Annals of Statistics*, *2*(6). https://doi.org/10.1214/aos/1176342874

Boomsma, W., Kent, J. T., Mardia, K. V., Taylor, C. C., & Hamelryck, T. (2006). Graphical models and directional statistics capture protein structure: 25th International Conference on Interdisciplinary Statistics and Bioinformatics. *Interdisciplinary Statistics and Bioinformatics*, 91–94.

Brust, M. R., & Strimbu, B. M. (2015). A Networked Swarm Model for UAV Deployment in the Assessment of Forest Environments. *2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, 1–6. https://doi.org/10.1109/ISSNIP.2015.7106967

Burrough, P. A., & McDonnell, R. A. (1998). *Principles of Geographical Information Systems*. Oxford University Press.

Cao, X., Liu, Z., Hu, C., Song, X., Quaye, J. A., & Lu, N. (2024). Three-Dimensional Geological Modelling in Earth Science Research: An In-Depth Review and Perspective Analysis. *Minerals*, *14*(7), 686. https://doi.org/10.3390/min14070686

Carmichael, T., & Ailleres, L. (2016). Method and analysis for the upscaling of structural data. *Journal of Structural Geology*, *83*, 121–133. https://doi.org/10.1016/j.jsg.2015.09.002

Carranza, E. J. M. (2009). Controls on mineral deposit occurrence inferred from analysis of their spatial pattern and spatial association with geological features. *Ore Geology Reviews*, *35*(3), 383–400. https://doi.org/10.1016/j.oregeorev.2009.01.001

Carta, J. A., Ramírez, P., & Bueno, C. (2008). A joint probability density function of wind speed and direction for wind energy analysis. *Energy Conversion and Management*, *49*(6), 1309–1320. https://doi.org/10.1016/j.enconman.2008.01.010

Caumon, G., Gray, G., Antoine, C., & Titeux, M.-O. (2013). Three-Dimensional Implicit Stratigraphic Model Building From Remote Sensing Data on Tetrahedral Meshes: Theory and Application to a Regional Model of La Popa Basin, NE Mexico. *IEEE Transactions on Geoscience and Remote Sensing*, *51*(3), 1613–1621. https://doi.org/10.1109/TGRS.2012.2207727

Chai, H., Orozco, L., Kannenberg, F., Siriwardena, L., Schwinn, T., Liu, H., Menges, A., & Yuan, P. F. (2024). Agent-Based Principal Strips Modeling for Freeform Surfaces in Architecture. *Nexus Network Journal*, *26*(2), 369–396. https://doi.org/10.1007/s00004-024-00765-0

Chandra Mohan, B., & Baskaran, R. (2012). A survey: Ant Colony Optimization based recent research and implementation on several engineering domain. *Expert Systems with Applications*, *39*(4), 4618–4627. https://doi.org/10.1016/j.eswa.2011.09.076

Chaplot, V., Darboux, F., Bourennane, H., Leguédois, S., Silvera, N., & Phachomphon, K. (2006). Accuracy of interpolation techniques for the derivation of digital elevation models in relation to landform types and data density. *Geomorphology*, *77*(1), 126–141. https://doi.org/10.1016/j.geomorph.2005.12.010

Cook, P. G., Banks, E. W., Marshall, S. K., Harrington, G. A., Batlle-Aguilar, J., Dogramaci, S., & Turnadge, C. (2022). Inferring fault hydrology using

groundwater age tracers. *Journal of Hydrology*, *610*, 127905. https://doi.org/10.1016/j.jhydrol.2022.127905

Cui, M., Hou, E., Lu, T., Hou, P., & Feng, D. (2025). Study on Spatial Interpolation Methods for High Precision 3D Geological Modeling of Coal Mining Faces. *Applied Sciences*, *15*(6), 2959. https://doi.org/10.3390/app15062959

Cuno, A., Esperança, C., Oliveira, A., & Cavalcanti, P. R. (2005). A hierarchical sampling approach for polygonizing variational implicit surfaces. *Computers & Graphics*, *29*(5), 676–685. https://doi.org/10.1016/j.cag.2005.08.005

Daniilidis, A., Saeid, S., & Doonechaly, N. G. (2021). The fault plane as the main fluid pathway: Geothermal field development options under subsurface and operational uncertainty. *Renewable Energy*, *171*, 927–946. https://doi.org/10.1016/j.renene.2021.02.148

de Kemp, E. A. (1999). Visualization of complex geological structures using 3-D Bézier construction tools. *Computers & Geosciences*, *25*(5), 581–597. https://doi.org/10.1016/S0098-3004(98)00159-9

De Kemp, E. A. (2007). 3-D geological modelling supporting mineral exploration. In *Geological Association of Canada, Mineral Deposits Division, Special Publication* (pp. 1051–1061). Geological Association of Canada. https://ostrnrcan-dostrncan.canada.ca/handle/1845/181198

de Kemp, E. A. (2021). Spatial agents for geological surface modelling. *Geoscientific Model Development*, *14*(11), 6661–6680. https://doi.org/10.5194/gmd-14-6661-2021

deKemp, E., Sprague, K., & Wong, W. (2004, November 1). *Interpretive Geology with Structural Constraints: An introduction to the SPARSE □ plug-in*. https://doi.org/10.5281/zenodo.4646210

Dimas, E., & Briassoulis, D. (1999). 3D geometric modelling based on NURBS: A review. *Advances in Engineering Software*, *30*(9), 741–751. https://doi.org/10.1016/S0965-9978(98)00110-0

Esri. (n.d.). *How inverse distance weighted interpolation works (ArcGIS Pro)*. Retrieved April 21, 2025, from https://pro.arcgis.com/en/pro-app/latest/help/analysis/geostatistical-analyst/how-inverse-distance-weighted-interpolation-works.htm

Fang, J.-J., & Hung, C.-L. (2013). An improved parameterization method for B-spline curve and surface interpolation. *Computer-Aided Design*, *45*(6), 1005–1028. https://doi.org/10.1016/j.cad.2013.01.005

Farin, G. (2002). 5—The Bernstein Form of a Bézier Curve. In G. Farin (Ed.), *Curves and Surfaces for CAGD (Fifth Edition)* (pp. 57–79). Morgan Kaufmann. https://doi.org/10.1016/B978-155860737-8/50005-3

Fitak, R. R., & Johnsen, S. (2017). Bringing the analysis of animal orientation data full circle: Model-based approaches with maximum likelihood. *The Journal of Experimental Biology*, *220*(21), 3878–3882. https://doi.org/10.1242/jeb.167056

Fleischer, M. (2005). *Foundations of Swarm Intelligence: From Principles to Practice* (No. arXiv:nlin/0502003). arXiv. https://doi.org/10.48550/arXiv.nlin/0502003

Gershenson, C., Trianni, V., Werfel, J., & Sayama, H. (2020). *Self-Organization and Artificial Life* (No. arXiv:1903.07456). arXiv. https://doi.org/10.48550/arXiv.1903.07456

González-Méndez, M., Olaya, C., Fasolino, I., Grimaldi, M., & Obregón, N. (2021). Agent-Based Modeling for Urban Development Planning based on Human Needs. Conceptual Basis and Model Formulation. *Land Use Policy*, *101*, 105110. https://doi.org/10.1016/j.landusepol.2020.105110

Goovaerts, P. (2000). Geostatistical approaches for incorporating elevation into the spatial interpolation of rainfall. *Journal of Hydrology*, *228*(1), 113–129. https://doi.org/10.1016/S0022-1694(00)00144-X

Grose, L., Laurent, G., Aillères, L., Armit, R., Jessell, M., & Caumon, G. (2017). Structural data constraints for implicit modeling of folds. *Journal of Structural Geology*, *104*, 80–92. https://doi.org/10.1016/j.jsg.2017.09.013

Gudmundsson, A. (2022). Transport of Geothermal Fluids along Dikes and Fault Zones. *Energies*, *15*(19), 7106. https://doi.org/10.3390/en15197106

Hemerijckx, L.-M., De Vos, K., Kaunda, J. O., & Van Rompaey, A. (2025). Future scenarios for urban agriculture and food security in sub-Saharan Africa: Modelling the urban land-food system in an agent-based approach. *Computers, Environment and Urban Systems*, *118*, 102258. https://doi.org/10.1016/j.compenvurbsys.2025.102258

Hilal, A., Bangroo, S. A., Kirmani, N. A., Wani, J. A., Biswas, A., Bhat, M. I., Farooq, K., Bashir, O., & Shah, T. I. (2024). Chapter 19—Geostatistical modeling—A tool for predictive soil mapping. In S. Lamine, P. K. Srivastava, A. Kayad, F.

Muñoz-Arriola, & P. C. Pandey (Eds.), *Remote Sensing in Precision Agriculture* (pp. 389–418). Academic Press. https://doi.org/10.1016/B978-0-323-91068-2.00011-4

Hillier, M., Schetselaar, E. M., de Kemp, E. A., & Perron, G. (2014). Three-Dimensional Modelling of Geological Surfaces Using Generalized Interpolation with Radial Basis Functions. *Mathematical Geosciences*, *46*(8), 931–953. https://doi.org/10.1007/s11004-014-9540-3

Hillier, M., Wellmann, F., Brodaric, B., de Kemp, E., & Schetselaar, E. (2021). Three-Dimensional Structural Geological Modeling Using Graph Neural Networks. *Mathematical Geosciences*, *53*(8), 1725–1749. https://doi.org/10.1007/s11004-021-09945-x

Hillier, M., Wellmann, F., de Kemp, E. A., Brodaric, B., Schetselaar, E., & Bédard, K. (2023). GeoINR 1.0: An implicit neural network approach to three-dimensional geological modelling. *Geoscientific Model Development*, *16*(23), 6987–7012. https://doi.org/10.5194/gmd-16-6987-2023

Jayr, S., Gringarten, E., Tertois, A. L., Mallet, J. L., & Dulac, J. C. (2008). The need for a correct geological modelling support: The advent of the UVT-transform. *First Break*, *26*(10). https://doi.org/10.3997/1365-2397.26.10.28558

Kent, J. T. (1982). The Fisher-Bingham Distribution on the Sphere. *Journal of the Royal Statistical Society: Series B (Methodological)*, *44*(1), 71–80.

Koldasbayeva, D., Tregubova, P., Gasanov, M., Zaytsev, A., Petrovskaia, A., & Burnaev, E. (2024). Challenges in data-driven geospatial modeling for environmental

research and practice. *Nature Communications*, *15*(1), 10700. https://doi.org/10.1038/s41467-024-55240-8

Krige, D. G. (1951). *A statistical approach to some basic mine valuation problems on the Witwatersrand*. *52*, 119–139.

Laurent, G., Ailleres, L., Grose, L., Caumon, G., Jessell, M., & Armit, R. (2016). Implicit modeling of folds and overprinting deformation. *Earth and Planetary Science Letters*, *456*, 26–38. https://doi.org/10.1016/j.epsl.2016.09.040

Li, J., & Heap, A. D. (2011). A review of comparative studies of spatial interpolation methods in environmental sciences: Performance and impact factors. *Ecological Informatics*, *6*(3), 228–241. https://doi.org/10.1016/j.ecoinf.2010.12.003

Li, J., & Heap, A. D. (2014). Spatial interpolation methods applied in the environmental sciences: A review. *Environmental Modelling & Software*, *53*, 173–189. https://doi.org/10.1016/j.envsoft.2013.12.008

Lin, M.-L., Chung, C.-F., Jeng, F.-S., & Yao, T.-C. (2007). The deformation of overburden soil induced by thrust faulting and its impact on underground tunnels. *Engineering Geology*, *92*(3), 110–132. https://doi.org/10.1016/j.enggeo.2007.03.008

Liu, Y., Dai, S., Zhou, Y., Ding, F., Li, M., Li, X., Zhao, Y., Guo, B., Li, T., & Han, J. (2024). Fault characteristics and their control on oil and gas accumulation in the southwestern Ordos Basin. *Energy Geoscience*, *5*(1), 100151. https://doi.org/10.1016/j.engeos.2022.100151

Lu, G. Y., & Wong, D. W. (2008). An adaptive inverse-distance weighting spatial interpolation technique. *Computers & Geosciences*, *34*(9), 1044–1055. https://doi.org/10.1016/j.cageo.2007.07.010

Luo, P., Wu, Y., & Song, Y. (2025). *Feature-free regression kriging* (No. arXiv:2507.07382). arXiv. https://doi.org/10.48550/arXiv.2507.07382

Mac Namee, B. (2012). Computer Graphics and Games, Agent Based Modeling in. In *Computational Complexity* (pp. 604–621). Springer, New York, NY. https://doi.org/10.1007/978-1-4614-1800-9_39

Maleika, W. (2020). Inverse distance weighting method optimization in the process of digital terrain model creation based on data collected from a multibeam echosounder. *Applied Geomatics*, *12*(4), 397–407. https://doi.org/10.1007/s12518-020-00307-6

Malik, F. H., & Lehtonen, M. (2016). A review: Agents in smart grids. *Electric Power Systems Research*, *131*, 71–79. https://doi.org/10.1016/j.epsr.2015.10.004

Mallet, J. L., Jacquemin, P., & Cheimanoff, N. (1989). GOCAD project: Geometric modeling of complex geological surfaces. In *SEG Technical Program Expanded Abstracts 1989* (pp. 126–128). Society of Exploration Geophysicists. https://doi.org/10.1190/1.1889515

Mallet, J.-L. (1992). Discrete smooth interpolation in geometric modelling. *Computer-Aided Design*, *24*(4), 178–191. https://doi.org/10.1016/0010-4485(92)90054-E

Mamur, H., Üstüner, M. A., & Bhuiyan, M. R. A. (2022). Future perspective and current situation of maximum power point tracking methods in thermoelectric generators.

Sustainable Energy Technologies and Assessments, *50*, 101824. https://doi.org/10.1016/j.seta.2021.101824

Maniteja, M., Samanta, G., Gebretsadik, A., Tsae, N. B., Rai, S. S., Fissha, Y., Okada, N., & Kawamura, Y. (2025). Advancing Iron Ore Grade Estimation: A Comparative Study of Machine Learning and Ordinary Kriging. *Minerals*, *15*(2), 131. https://doi.org/10.3390/min15020131

Mardia, K. V., & Jupp, P. E. (Eds.). (2010). *Directional statistics*. J. Wiley.

Matheron, G. (1963). Principles of geostatistics. *Economic Geology*, *58*(8), 1246–1266. https://doi.org/10.2113/gsecongeo.58.8.1246

McMechan, M. E. (2013). *Geology, Kananaskis Lakes, British Columbia—Alberta* (p. 11). Natural Resources Canada. https://doi.org/10.4095/288953

Moghaddas Tafreshi, S. M., Ranjbarzadeh, H., Jafari, M., & Khayyam, H. (2016). A probabilistic unit commitment model for optimal operation of plug-in electric vehicles in microgrid. *Renewable and Sustainable Energy Reviews*, *66*, 934–947. https://doi.org/10.1016/j.rser.2016.08.013

Odeh, I. O. A., McBratney, A. B., & Chittleborough, D. J. (1995). Further results on prediction of soil properties from terrain attributes: Heterotopic cokriging and regression-kriging. *Geoderma*, *67*(3), 215–226. https://doi.org/10.1016/0016-7061(95)00007-B

Okdem, S., & Karaboga, D. (2009). Routing in Wireless Sensor Networks Using an Ant Colony Optimization (ACO) Router Chip. *Sensors*, *9*(2), 909–921. https://doi.org/10.3390/s90200909

Paine, P. J., Preston, S. P., Tsagris, M., & Wood, A. T. A. (2018). An elliptically symmetric angular Gaussian distribution. *Statistics and Computing*, *28*(3), 689–697. https://doi.org/10.1007/s11222-017-9756-4

Paudel, J. (2025). Economic impact of large earthquakes: Lessons from residential property values. *Oxford Economic Papers*, *77*(2), 564–583. https://doi.org/10.1093/oep/gpae041

Peng, N., Wang, Z., Nie, B., Zhang, X., & Liu, Y. (2023). Correlation between fractal characteristics of fault structures and metallogenic density in the Nanling area, South China. *Ore Geology Reviews*, *162*, 105717. https://doi.org/10.1016/j.oregeorev.2023.105717

Pewsey, A., & García-Portugués, E. (2020). *Recent advances in directional statistics* (No. arXiv:2005.06889). arXiv. https://doi.org/10.48550/arXiv.2005.06889

Piegl, L. (1991). On NURBS: A survey. *IEEE Computer Graphics and Applications*, *11*(1), 55–71. https://doi.org/10.1109/38.67702

Pyrcz, M. J., & Deutsch, C. V. (2014). *Geostatistical Reservoir Modeling* (2nd ed.). Oxford University Press.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics*, *21*(4), 25–34. https://doi.org/10.1145/37402.37406

Said Mad Zain, S. A. A. A., Misro, M. Y., & Miura, K. T. (2023). Generalized Riemann-Liouville fractional Bézier curve and its applications in engineering surface. *Alexandria Engineering Journal*, *65*, 585–606. https://doi.org/10.1016/j.aej.2022.10.044

Schirmer, L., Novello, T., da Silva, V., Schardong, G., Perazzo, D., Lopes, H., Gonçalves, N., & Velho, L. (2024). Geometric implicit neural representations for signed distance functions. *Computers & Graphics*, *125*, 104085. https://doi.org/10.1016/j.cag.2024.104085

Sengupta, S., Basak, S., & II, R. A. P. (2018). Particle Swarm Optimization: A survey of historical and recent developments with hybridization perspectives. *Machine Learning and Knowledge Extraction*, *1*(1), 157–191. https://doi.org/10.3390/make1010010

Shepard, D. (1968). A two-dimensional interpolation function for irregularly-spaced data. *Proceedings of the 1968 23rd ACM National Conference On* -, 517–524. https://doi.org/10.1145/800186.810616

Shi, Y., & Eberhart, R. C. (1998). Parameter selection in particle swarm optimization. In V. W. Porto, N. Saravanan, D. Waagen, & A. E. Eiben (Eds.), *Evolutionary Programming VII* (pp. 591–600). Springer. https://doi.org/10.1007/BFb0040810

Sousa, D. S., Silva, E. P., Alves, C. de M. A., Minoti, R. T., & Vergara, F. E. (2025). Coupling data-driven agent-based and hydrological modelling to explore the effect of collective water allocation strategies in environmental flows. *Journal of Hydrology*, *652*, 132670. https://doi.org/10.1016/j.jhydrol.2025.132670

Sprague, K. B., & de Kemp, E. A. (2005). Interpretive Tools for 3-D Structural Geological Modelling Part II: Surface Design from Sparse Spatial Data. *GeoInformatica*, *9*(1), 5–32. https://doi.org/10.1007/s10707-004-5620-8

Stasinchuk, Y., Vrba, M., Petrlík, M., Báča, T., Spurný, V., Hert, D., Žaitlík, D., Nascimento, T., & Saska, M. (2021). A Multi-UAV System for Detection and

Elimination of Multiple Targets. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 555–561. https://doi.org/10.1109/ICRA48506.2021.9562057

Tang, X., Yu, F., & Chen, R. (2010). Path planning of underwater vehicle based on particle swarm optimization. *Proc. Int. Conf. Intelligent Control Inf. Process., ICICIP*, *PART 1*, 123–126. https://doi.org/10.1109/ICICIP.2010.5564218

Templeton, A., Xie, H., Gwynne, S., Hunt, A., Thompson, P., & Köster, G. (2024). Agent-based models of social behaviour and communication in evacuations: A systematic review. *Safety Science*, *176*, 106520. https://doi.org/10.1016/j.ssci.2024.106520

Testa, A., Boncio, P., Pace, B., Mirabella, F., Pauselli, C., Ercoli, M., Auciello, E., Visini, F., & Baize, S. (2024). Probabilistic fault displacement Hazard analysis in an extensional setting: Application to a strategic Dam and methodological implications. *Engineering Geology*, *343*, 107817. https://doi.org/10.1016/j.enggeo.2024.107817

Thapa, P., & McMechan, M. E. (2019). *Methodology for portraying 3D structure using ArcGIS: A test case from the southern Canadian Rocky Mountains, British Columbia and Alberta* (No. 8576; p. 8576). https://doi.org/10.4095/314941

Tobler, W. R. (1970). A Computer Movie Simulating Urban Growth in the Detroit Region. *Economic Geography*, *46*, 234. https://doi.org/10.2307/143141

Trauth, M. H. (2007). Statistics on Directional Data. In M. H. Trauth (Ed.), *MATLAB® Recipes for Earth Sciences* (pp. 263–277). Springer. https://doi.org/10.1007/978-3-540-72749-1_10

Turrell, T. (2016, December 16). *Agent-based models: Understanding the economy from the bottom up*. Bank of England. https://www.bankofengland.co.uk/quarterly-bulletin/2016/q4/agent-based-models-understanding-the-economy-from-the-bottom-up

U.S. Geological Survey. (2025). *What is a fault and what are the different types?* U.S. Geological Survey (USGS). https://www.usgs.gov/faqs/what-a-fault-and-what-are-different-types

U.S. Geological Survey; Federal Emergency Management Agency. (2023). *New USGS–FEMA study highlights economic earthquake risk in the United States* [Technical Report]. U.S. Geological Survey. https://www.usgs.gov/news/national-news-release/new-usgs-fema-study-highlights-economic-earthquake-risk-united-states?utm_source=chatgpt.com

Wagner, P. D., Fiener, P., Wilken, F., Kumar, S., & Schneider, K. (2012). Comparison and evaluation of spatial interpolation schemes for daily rainfall in data scarce regions. *Journal of Hydrology*, *464–465*, 388–400. https://doi.org/10.1016/j.jhydrol.2012.07.026

Webster, R. (2005). SPATIAL VARIATION, SOIL PROPERTIES. In D. Hillel (Ed.), *Encyclopedia of Soils in the Environment* (pp. 1–13). Elsevier. https://doi.org/10.1016/B0-12-348530-4/00418-5

Wellmann, F., & Caumon, G. (2018). Chapter One - 3-D Structural geological models: Concepts, methods, and uncertainties. In C. Schmelzbach (Ed.), *Advances in Geophysics* (Vol. 59, pp. 1–121). Elsevier. https://doi.org/10.1016/bs.agph.2018.09.001

Yuan, T. (2021). The 8-parameter Fisher–Bingham distribution on the sphere. *Computational Statistics*, *36*(1), 409–420. https://doi.org/10.1007/s00180-020-01023-w

Zhan. (2024, September 26). *7,598 drones set new world record with stunning aerial display*. Guinness World Records. https://www.guinnessworldrecords.com/news/commercial/2024/9/7598-drones-set-new-world-record-with-stunning-aerial-display.html

Zhou, Q., & Li, J. (2020). Geo-Spatial Analysis in Hydrology. *ISPRS International Journal of Geo-Information*, *9*(7), 435. https://doi.org/10.3390/ijgi9070435

Appendix A

Structural agents' terms and model parameters

The terms used to describe structural agents, including their description, are summarized in Table A1.

Table A1 Terms and abbreviations

Term (Abbreviation)	Description
Structural Agent (SA)	Agent with spatial location (X,Y,Z), Normal (N1,N2,N3) and Velocity (V1,V2,V3)
Swarm	Connected group of structural agents moving together based on flocking rules.
Flockmates	Agents within perception radius and angular tolerance whose positions and velocities influence the focal agent.
Neighbour	Any agent or data point within perception radius; may be excluded as a flockmate if angular tolerance is exceeded.
Spherical Linear Interpolation (SLERP)	Smooth rotation interpolation between two orientations using quaternions.
Eigen-modulated force	Force scaled along local eigenvectors from PCA of neighbouring normals.
Strike	Compass direction of the line formed by the intersection of geological surface with a horizontal plane
Dip	Angle between the geological surface and a horizontal plane, measure perpendicular to the strike direction

The parameters used in the model, as well as their default values (unless changed), are presented in Table A2 below.

Table A2 Model parameters used in the simulation

Parameter Name	Description	Default Value	Application
alignment_weight	Influence of neighbours' direction on agent orientation	0.03	Global
cohesion_weight	Attraction force strength toward neighbouring agents	0.1	Global
separation_weight	Repulsion force to avoid crowding with neighbours	0.1	Global
speed	Base movement speed of agents	2	Global
mini_swarm_size	Minimum number of agents required to form a swarm	30	Global
angle_align_tolerance	Max angle (°) between normals to be considered aligned	60	Global
space_margin	Extra margin added around the data bounding box	100	Global
separation_distance	Minimum allowed distance between agents	0.016 × space diagonal	Global
perception_distance	Max distance for agent to perceive neighbours	0.16 × space diagonal	Global
scaling_factor	Scale factor for glyph visualisation	0.0003165 × space diagonal	Global
w_inertia	Inertia weight for PSO updates	0.5	PSO
w_cogn	Cognitive weight for PSO updates	1.5	PSO
w_social	Social weight for PSO updates	1.5	PSO
pso_iteration	Max PSO iterations per simulation step	100	PSO
PSO tolerence	Convergence tolerance between two consecutive steps	1.00E-04	PSO
PSO patience	Number of steps required to declare convergence	20	PSO
normal_clip	Clipping threshold for displaying normal vectors	30	Global
w_radial	Weight for staying within desired spawn radius	5	PSO
w_surface	Weight for being close to the local fitting plane	10	PSO
w_data_sep	Penalty for spawning too close to data	5	PSO
w_plane	Penalty for deviating from parent plane	1	PSO
w_kent	Weight for Kent distribution likelihood in normal estimation	1	PSO
angle_tolerance	Max deviation from parent normal for PSO candidate	10	PSO
attraction_weights	Directional weights for attraction force (PCA axes)	[0.1, 0.1, 0.9]	SwarmControl
repulsion_weights	Directional weights for repulsion force (PCA axes)	[0.9, 0.9, 0.1]	SwarmControl

Appendix B – Libraries and code examples

I. Eigen decomposition and spatial biasing

The code for eigen decomposition used to get the local plane normal (p3) in order to bias agents to be attracted to a flat world is presented below

```
def eigen_vector(self, normals, neighbours):
    if len(neighbours) < 2:
        p3 = np.array([0.0, 0.0, 1.0])
        p2 = np.array([0.0, 1.0, 0.0])
        p1 = np.array([1.0, 0.0, 0.0])

    else:
        neighbor_normals = normals[neighbours]
        covariance_mtrx = np.cov(neighbor_normals, rowvar=False)
        eigenvalues, eigenvectors = np.linalg.eig(covariance_mtrx)
        eigenvalues = np.real(eigenvalues)
        eigenvectors = np.real(eigenvectors)

        sorted_values = np.argsort(eigenvalues) #  from highest variance to lowest
        p3 = eigenvectors[:, sorted_values[0]]  # Least eigenvalue (smallest variance direction)
        p2 = eigenvectors[:, sorted_values[1]]  # Second largest eigenvalue
        p1 = eigenvectors[:, sorted_values[2]]  # Largest eigenvalue (greatest variance direction)"""

    # Normalize the vectors
    p3 /= np.linalg.norm(p3)
    p2 /= np.linalg.norm(p2)
    p1 /= np.linalg.norm(p1)

    return np.array([p1, p2, p3])
def eigen_effect(self, p1, p2, p3, force, array):
    # Apply the array weights to each principal direction's contribution
    influence_from_p1 = array[0] * np.dot(force, p1) * p1
    influence_from_p2 = array[1] * np.dot(force, p2) * p2
    influence_from_p3 = array[2] * np.dot(force, p3) * p3

    # Sum the weighted contributions
    total_influenced_force = influence_from_p1 + influence_from_p2 + influence_from_p3

    return total_influenced_force
```

II. Spatial KD Tree

A k-d tree is a data structure that organizes points in k-dimensional space for efficient searching. It works by recursively splitting the data along coordinate axes to form a balanced binary tree. This allows rapid nearest-neighbour and range queries compared to checking every point directly. An example of neighbour querying within a given swarm using KD Tree is shown below. Complete details can be found at: https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.KDTree.html

```
for i in swarm_indices:
    force_vector = np.zeros(3)

    # build KDTree and find neighbours in combined space
    tree = cKDTree(all_positions)
    neighbor_radius = self.boid_inst.perception_distance
    local_i = local_indices_map[i]
    query_pos = positions_swarm[local_i]
    neighbours = tree.query_ball_point(query_pos, r=neighbor_radius)
```

III. NetworkX

NetworkX is a Python library for creating, analyzing, and visualizing graphs.
It represents data as nodes (points) connected by edges (links), allowing easy modelling of relationships. It is used it to store and process spatial connections between agents or data points, enabling operations like finding neighbours, shortest paths, and connected components. In this paper it served as a way to build connected components of agents, identifying them as a swarm, as illustrated in the figure below. More details can be found at: https://networkx.org/

```python
def detect_swarms(self, distance_threshold, min_swarm_size, angle_threshold=60):
    """
    Swarms are computed based on connected components using a distance threshold.
    Each boid is connected to any other boid within the distance threshold.
    Only components with size >= min_swarm_size are considered as swarms.
    """
    # Build a KDTree for the current boid positions
    tree = cKDTree(self.positions)

    # Create an empty graph and add all boid indices as nodes.
    G = nx.Graph()
    num_boids = self.positions.shape[0]
    G.add_nodes_from(range(num_boids))

    # For each boid, find neighbors within the distance_threshold and add edges.
    for i in range(num_boids):
        indices = tree.query_ball_point(self.positions[i], r=distance_threshold)
        normal_i = self.normals[i] / (np.linalg.norm(self.normals[i]) + 1e-5)

        for j in indices:
            if i == j:
                continue

            normal_j = self.normals[j] / (np.linalg.norm(self.normals[j]) + 1e-5)
            dot_product = np.dot(normal_i, normal_j)
            angle = np.degrees(np.arccos(np.clip(dot_product, -1.0, a_max: 1.0)))

            if angle <= angle_threshold:
                G.add_edge(i, j)

    # Identify connected components that are large enough to be considered swarms.
    swarms = [list(component) for component in nx.connected_components(G) if len(component) >= min_swarm_size]

    # All boids not part of any valid swarm will be in the non-swarm list.
    swarm_indices = set(i for swarm in swarms for i in swarm)
    non_swarm = [i for i in range(num_boids) if i not in swarm_indices]

    return swarms, non_swarm
```

IV. Quaternion rotation

Quaternions offer a way to represent 3D rotations without having the gimbal lock effect, common in Euler angles. A rotation matrix R can be constructed from quaternion components q_w,q_x,q_y,q_z (de Kemp, 2021):

$$R = \begin{bmatrix} q2w + q2x - q2y - q2z & 2qxqy - 2qzqw & 2qzqx + 2qyqw \\ 2qxqy + 2qzqw & q2wq2x + q2y - q2z & 2qyqz - 2qwqx \\ 2qzqx - 2qyqw & 2qyqz + 2qwqx & q2wq2x - q2y - q2z \end{bmatrix} \quad (9)$$

where:

$q2w = 1-qx^2-qy^2-qz^2$

$q2x = qx^2$

$q2y = qy^2$

$q2z = qz^2$

In this paper, Rotation library from SciPy was used apply quaternion-based rotations efficiently in 3D space

```
def quaternion_rotation(self, vector, angle, axis):
    # Normalize the axis
    axis_normalized = axis / (np.linalg.norm(axis) + 1e-5)

    # Create quaternion using axis-angle representation
    rotation = R.from_rotvec(angle * axis_normalized)

    # Apply rotation
    rotated_vector = rotation.apply(vector)

    return rotated_vector
```

V. Spherical Linear Interpolation (SLERP)

SLERP is a method for smoothly interpolating between two orientations represented as quaternions. It treats the quaternions as points on the unit 4D sphere and finds the shortest arc between them, interpolating along this arc at a constant angular speed.
This ensures smooth, uniform rotation without distortion, making it ideal for blending agent orientations over time. In this study, the current and target direction vectors are first normalized and converted into rotation representations. SLERP is then applied to interpolate between these orientations, and the result is converted back to a rotation vector for use in the agent update.

```python
def apply_slerp_rotation(self, current_vector, target_vector, t=0.8):
    """Applying SLERP between current_vector and target_vector using SciPy's Slerp"""

    # Normalize the vectors
    current_vector = current_vector / (np.linalg.norm(current_vector) + 1e-5)
    target_vector = target_vector / (np.linalg.norm(target_vector) + 1e-5)

    # Create rotations from vectors
    current_rot = R.from_rotvec(current_vector)
    target_rot = R.from_rotvec(target_vector)

    # Perform SLERP interpolation
    slerp = Slerp( *args: [0, 1], R.from_rotvec([current_rot.as_rotvec(), target_rot.as_rotvec()]))
    interpolated_rot = slerp([t])  # Interpolate at time t

    # Convert the interpolated rotation back to a vector
    return interpolated_rot[0].as_rotvec()
```

VI. Selecting a neighbour

Neighbours are selected by building a KD tree and finding agents within the perception radius in a Euclidean space. Then, an additional angular constraint is imposed to filter agents whose normal orientations are very different from the focal agent. This way agents with nearly similar attributes are considered as neighbours to avoid instability.

```python
def _filter_neighbors(self,
                      positions: np.ndarray,normals: np.ndarray, center_pos: np.ndarray,center_norm: np.ndarray,
                      min_dist: float = 1e-5,max_dist: float = None, angle_tolerance: float = None,
                      k: int = None
                      ) -> np.ndarray:
    """
    Return the indices of up to k candidates whose distance to center_pos
    is within (min_dist, max_dist) and whose normal makes an angle <= angle_tolerance
    with center_norm. If k is None, return all that pass the filters.
    """
    diff = positions - center_pos
    dists = np.linalg.norm(diff, axis=1)
    mask = dists > min_dist

    if max_dist is not None:
        mask &= (dists < max_dist)

    if angle_tolerance is not None:
        dots = np.clip(normals @ center_norm, -1.0,  a_max: 1.0)
        angles = np.degrees(np.arccos(dots))
        mask &= (angles <= angle_tolerance)

    idxs = np.nonzero(mask)[0]
    if k is not None and len(idxs) > k:
        # pick the k smallest distances among those
        nearest_order = np.argsort(dists[idxs])[:k]
        idxs = idxs[nearest_order]

    return idxs
```

VII. Updating agent's normal and velocities simultaneously

To ensure an agent's velocity always remains on its plane, thus orthogonal to its normal orientation, when one is updated, the other one is updated at the same time.

```python
def update_normals_and_velocities(self, velocity, desired_velocity, normal):
    """Ensuring agent's normal remains perpendicular to its velocity"""
    # Normalize the velocity vectors
    velocity_norm = velocity / (np.linalg.norm(velocity) + 1e-5)
    desired_velocity_norm = desired_velocity / (np.linalg.norm(desired_velocity) + 1e-5)
    normal_norm = normal / (np.linalg.norm(normal) + 1e-5)

    # Calculate the angle and axis for quaternion rotation
    cos_angle = np.dot(velocity_norm, desired_velocity_norm)
    angle = np.arccos(np.clip(cos_angle, -1, a_max: 1))
    axis = np.cross(velocity_norm, desired_velocity_norm)

    # Apply the same quaternion rotation to both velocity and normal
    new_velocity = self.quaternion_rotation(velocity_norm, angle, axis)
    new_normal = self.quaternion_rotation(normal_norm, angle, axis)
    new_normal = new_normal / (np.linalg.norm(new_normal) + 1e-5)

    return new_velocity, new_normal
```

Appendix C – User Guide

The app can be installed directly from the GitHub repository. By following the following steps

- Open a terminal and run: git clone https://github.com/Roger-Niyongira/Structural-agent-based-model
- cd into Structural Agent Based Model
- create a new virtual environment
- Install the requirements.txt
- Now that the app can be launched, open GUI_Agents.py, which is where all the models are controlled and ensure the main is the selected git branch.

To see the model running without any data, simply click on the simulate button (left panel). A message reminding that there is no data to be used will display, click OK or close the message window. At this stage, agents can be observed without any data constraints. Change flocking weights to experiment with different shapes. Once the app is closed and reopened, everything returns to default. It is possible to explore different behaviours by opening NoDataABM.py and in the update method, replace influenced attraction/ repulsion by attraction and repulsion in force_vector computation.

```
for j in range(self.num_boids):
    if i != j:
        distance = self.calculate_distance(i, j)
        # Calculate repulsion and attraction forces based on the distance
        repulsion = self.calculate_repulsion_vector(i, j, distance)
        attraction = self.calculate_attraction_vector(i, j, distance)
        influenced_attraction = self.eigen_effect(p1, p2, p3, attraction, self.attraction_weights)
        influenced_repulsion = self.eigen_effect(p1, p2, p3, repulsion, self.repulsion_weights)
        alignment = self.calculate_alignment_vector(i)

        # Combine forces
        force_vector += (self.cohesion_weight * influenced_attraction
                         + self.separation_weight * influenced_repulsion
                         + self.alignment_weight * alignment)
```

For the model with data, valid dataset must be loaded first. If trace points are marked as data, they need at least another dataset with orientation information to be able to run. Then, click simulate to show the starting conditions (before any agent is introduced). A placeholder agent maybe observed, but it has no purpose at this stage and will be removed in the first few steps. This is because the model was developed with the intention that at least one agent should be present at all times, and to allow visualisation of no agent, the number of agents is set to 1.

As the agent population grows, it is recommended to pause the simulation before rotating the camera or screen view, otherwise there will be a delay due to heavy computations happening in the background.