Mathematical Foundations of Prioritizing Shoggoths: A Formal Framework for Distributed Semantic Attention

Flyxion

July 27, 2025

Abstract

This essay presents a comprehensive mathematical framework for *Prioritizing Shoggoths*, recursive, distributed attention mechanisms that scan multimodal data (text–audio–images–3D models) for relevance to evolving priority vectors. Integrating the original Relativistic Scalar Vector Plenum (RSVP) theory, formalized through an OWL ontology, Basic Formal Ontology (BFO), sheaf theory, category theory, Karl Fant's NULL Convention Logic (NCL), and the Ontological Polycompiler and Yarncrawler Framework metaphors, Shoggoths operate as clockless, concurrent agents. Enhanced with photogrammetry, OCR/ASR correction, 3D anchoring, and error memory, the framework supports semantic reconstruction and real-world error detection for applications in anomaly detection, code synthesis, robotics, and AR/VR. It leverages category-theoretic functors, BFO grounding, sheaf logic, RSVP fields, NCL asynchrony, and an ergodic edge network scheduler. Innovative implications span neurosymbolic AI, ethical AI, quantum-inspired computing, and bio-inspired systems, with comparisons to traditional algorithms, Fant's model, and biological systems. Appendices provide formal definitions, RSVP equations, OWL encodings, and prototype code.

1 Introduction

The rapid evolution of artificial intelligence (AI) demands computational paradigms that transcend traditional sequential architectures to address distributed, relevance-aware, and ethically bounded systems capable of processing complex, multimodal data–text, audio, images, and 3D models. *Prioritizing Shoggoths* are recursive, semi-autonomous agents designed to scan data sources–text, audio, images, and 3D models–for alignment with dynamic priority vectors, inspired by the Shoggoth metaphor: a complex, adaptive entity with layered attention mechanisms. This framework unifies the original Relativistic Scalar Vector Plenum (RSVP) theory, formalized through an OWL ontology defining scalar, vector, and entropy fields; Basic Formal Ontology (BFO) for structured grounding; sheaf and category theory for mathematical rigor; and Karl Fant's NULL Convention Logic (NCL) for clockless concurrency (Fant, 2005, 2007). Enhanced with photogrammetry, OCR/ASR correction, 3D anchoring, and a global error ledger, Shoggoths support applications like AR/VR and robotics. The

Ontological Polycompiler Paradigm fuses disparate ontologies, while the Yarncrawler Framework navigates semantic threads, repairing entropy-driven drift. Contributions include categorical attention modeling, ontological grounding, sheaf-theoretic error correction, asynchronous edge computing, and innovative implications for neurosymbolic AI, ethical AI, quantum-inspired computing, and bio-inspired systems, addressing key challenges in distributed AI as of July 27, 2025.

2 Prerequisites

Engaging with the Prioritizing Shoggoth framework requires a robust foundation in advanced mathematical and computational disciplines. Readers must be familiar with category theory, including categories, functors, natural transformations, and colimits, which underpin Shoggoths' abstract mappings (Lawvere and Schanuel, 2009). Sheaf theory, encompassing presheaves, sheaves, and cohomology, ensures local-to-global data coherence (Bredon, 1997). The original RSVP theory, formalized via an OWL ontology, models cognitive dynamics through scalar (Φ), vector ($\mathbf{v}$), and entropy ($\mathcal{S}$) fields, driving attention mechanisms. Basic Formal Ontology (BFO) distinguishes continuants from occurrents, grounding computations (Arp et al., 2015). Karl Fant's NCL enables asynchronous processing (Fant, 2005, 2007). These prerequisites equip readers to navigate the framework's theoretical depth and practical applications in processing data–text, audio, images, and 3D models.

3 Background Concepts

The Prioritizing Shoggoth framework integrates a rich set of theoretical constructs to enable distributed semantic attention across multimodal data–text, audio, images, and 3D models. This section introduces the RSVP theory, formalized through an OWL ontology; BFO for ontological structure; category and sheaf theory for rigor; and Fant's NCL for concurrency. The Ontological Polycompiler and Yarncrawler Framework metaphors conceptualize Shoggoths as ontology-fusing, thread-repairing agents. These components enable Shoggoths to process complex data environments, supporting applications in robotics, AR/VR, and ethical AI.

3.1 RSVP Field Theory

The Relativistic Scalar Vector Plenum (RSVP) theory, an original contribution, models cognitive dynamics through three fields: a scalar field (Φ) for attentional density, a vector field ($\mathbf{v}$) for semantic attractors, and an entropy field ($\mathcal{S}$) for cognitive tension. Dynamics are:

$$\frac{d\Phi}{dt} = -\alpha(\Phi - \Phi_{\text{target}}), \quad \frac{d\mathcal{S}}{dt} = \beta(\text{novelty} - \text{resolution}), \quad \partial_t \mathbf{v} + (\mathbf{v} \cdot \nabla)\mathbf{v} = -\nabla\Phi - \nabla\mathcal{S}.$$

The OWL ontology (`RSVP-ontology.owl`) defines classes like `RSVP-Scalar-Field` and processes like `Lamphrodyne-Coupling`, grounding dynamics in BFO (Arp et al., 2015). Negentropy flows stabilize attention, enabling Shoggoths to prioritize data and detect patterns.

3.1.1 Worked Example: Text Anomaly Detection

In a 10,000-line corpus, a Shoggoth detects teh (vs. the). High Φ spreads attention; an entropy spike ($\mathcal{S} > \epsilon$) at teh focuses $\mathbf{v}$, lowering Φ. An LLM corrects the error, reducing $\mathcal{S}$.

3.1.2 Ontological Grounding

The OWL ontology defines: - `RSVP-Scalar-Field`: Inheres in spacetime regions, participates in evolution processes. - `Lamphrodyne-Coupling`: Couples $\mathbf{v}$ and $\mathcal{S}$, resulting in smoothing. - `Entropic-Redshift`: Dissipates entropy, causing energy shifts.

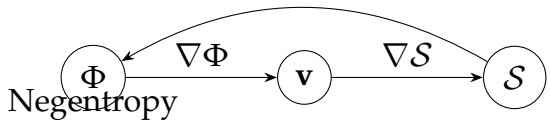

Figure 1: RSVP Field Interactions

3.2 Basic Formal Ontology (BFO)

BFO distinguishes *continuants* (e.g., priority vectors) from *occurrents* (e.g., scans) (Arp et al., 2015), grounding Shoggoths in a structured reality.

3.3 Category Theory and Sheaf Logic

Category theory defines mappings (Lawvere and Schanuel, 2009). Sheaves assign data to open sets, with cohomology ($H^1(F) \neq 0$) detecting errors (Bredon, 1997).

3.4 NULL Convention Logic and Invocation Model

Fant's *Computer Science Reconsidered* proposes an invocation model via NCL (Fant, 2007, 2005), enabling clockless concurrency.

3.5 Ontological Polycompiler and Yarncrawler Framework

Polycompilers fuse ontologies ($\mathcal{C}_i : \mathcal{O}_i \to \mathcal{U}$, $\varinjlim \mathcal{C}_i$) with RSVP embeddings ($\mathcal{O}_i \hookrightarrow \mathrm{Open}(X_{\mathrm{RSVP}})$). Yarncrawlers traverse semantic spaces ($\gamma : X_i \to X_j$), repairing drift ($\partial\mathcal{S}/\partial t > \epsilon$) with NCL logic ($\Box_{\mathrm{null}}\phi$).

3.5.1 Mathematical Mappings

4 Theoretical Construction of the Prioritizing Shoggoth

This section formalizes Shoggoths as recursive agents scanning multimodal data–text, audio, images, and 3D models–leveraging category theory, RSVP, BFO, NCL, and Polycompiler/Yarncrawler metaphors. Shoggoths embed data, trigger alerts, and operate asynchronously, supporting applications like AR/VR and robotics.

Concept	Polycompiler	Yarncrawler
Structure	Functor $\mathcal{C}_i : \mathcal{O}_i \to \mathcal{U}$	Morphism $\gamma : X_i \to X_j$
Operation	2-Colimit $\varinjlim \mathcal{C}_i$	Homotopy retraction
Error Detection	Cohomology $H^1(F)$	Entropy spike $\partial\mathcal{S}/\partial t > \epsilon$
Logic	Sheaf gluing	NCL ($\Box_{\text{null}}\phi$)

Table 1: Polycompiler and Yarncrawler Mappings

4.1 Semantic Trigger Structure

Shoggoths scan $\mathcal{D}$ (text–audio–images):

$$E : \mathcal{D} \to \mathcal{V}ec, \quad E(d) \in \mathbb{R}^n.$$

Similarity:

$$\text{Sim} : \mathcal{V}ec^{\text{op}} \times \mathcal{V}ec \to \mathbb{R}, \quad \text{Sim}(v, w) = \frac{\langle v, w \rangle}{\|v\| \|w\|}.$$

Trigger:

$$\text{Trig}_\lambda(v, p) = \begin{cases} 1 & \text{if } \text{Sim}(v, p) > \lambda, \\ 0 & \text{otherwise.} \end{cases}$$

Shoggoth functor:

$$\mathcal{S} : \mathcal{D} \to \text{Alerts}, \quad \mathcal{S}(d) = \text{Trig}_\lambda(E(d), p).$$

4.1.1 Worked Example: Email Classification

A Shoggoth scans emails for spam. Priority vectors p_1, p_2 represent spam and urgent. An email win free money yields v with $\text{Sim}(v, p_1) = 0.9 > \lambda = 0.85$, triggering a spam alert via NCL.

4.2 RSVP Ontological Encoding

The $\mathcal{RSVP}$ category, grounded by `RSVP-ontology.owl`, maps fields to BFO:

$$\mathcal{BFO} \to \mathcal{RSVP}.$$

4.3 NCL Integration

NCL enables asynchronous triggers (Fant, 2005):

$$\mathcal{N}(d) = \top \iff \exists p \in \mathcal{P} : \text{Trig}_\lambda(E(d), p) = 1.$$

4.4 Ontology-Driven Coordination

The OWL ontology ensures Shoggoths align triggers with RSVP fields, using `Lamphrodyne-Coupling` to smooth entropy gradients.

5 Alternating Line Completion and Entropic Dropout

Training Shoggoths reconstructs missing data–text or code–using semantic distance, RSVP fields, and NCL. Polycompilers fuse ontologies, and Yarncrawlers repair drift, supporting code synthesis and text completion.

5.1 Training Corpus as a Fibered Category

Alternating completion drops segments:

$$A : C_i \to C_i'.$$

Shoggoths interpolate:

$$\mathcal{S} : \{C_i'\} \to \widehat{C}_i,$$

using:

$$\mathrm{Dist}(C_i', \widehat{C}_i) = \min_{\text{model}} \mathrm{Sim}(E(C_i'), E(\widehat{C}_i)).$$

Corrections:

$$w_{\text{new}} = w_{\text{old}} + \eta \cdot \text{ErrorGradient}.$$

5.1.1 Worked Example: Code Synthesis

A Shoggoth completes a Python script, fusing syntax and semantics via Polycompilers.

5.2 Homotopy Interpretation

Completions are homotopies:

$$H : C_i' \times [0,1] \to \widehat{C}_i.$$

5.3 Comparison with Fant's Invocation Model

Shoggoths mirror NCL's concurrency (Fant, 2007).

6 Edge Network as a Sheaf-Theoretic Compute Topology

The ergodic edge network scheduler distributes tasks using SSH and NCL (Fant, 2005). Yarncrawlers navigate node interactions, and Polycompilers fuse ontologies, enhancing scalability.

6.1 Category of Nodes

Nodes in $\mathcal{N}$:

$$F_x : \mathrm{Open}(X_x) \to \mathcal{V}ec.$$

Global sheaf:

$$F = \varinjlim_{x \in \mathcal{N}} F_x.$$

Discovery:

```
while read host; do
  ssh -o ConnectTimeout=5 "$host" "uptime; df -h; nproc; free -m" &
done < hosts.txt
```

Cost:

$$\text{Cost}(\text{node}) = w_1 \cdot \text{TransferTime} + w_2 \cdot \text{ComputeLatency} + w_3 \cdot \text{AccessPenalty}.$$

6.2 NCL-Driven Asynchrony

NCL enables clockless coordination (Fant, 2005).

6.3 Comparison with Traditional Networks

Unlike Lamport clocks (Lamport, 1978), Shoggoths use NCL

7 Multimodal Data Calibration and Obstruction Detection

Shoggoths handle multimodal data–text, audio, images, 3D frames–through photogrammetry, OCR/ASR correction, 3D anchoring, and error memory, with Polycompilers and Yarncrawlers supporting AR/VR and robotics.

7.1 Multimodal Category $\mathcal{M}$

Images, 3D frames, text–audio are processed via COLMAP.

7.1.1 Worked Example: Storefront Sign Correction

Images of Coffe Shop yield a 3D mesh. Tesseract extracts Coffe, corrected to Coffee with coordinates (x, y, z) for AR.

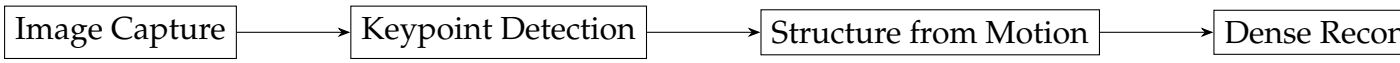

Figure 2: Photogrammetry Pipeline

7.2 Semantic Sheaf $F_{\textbf{multi}}$

Errors detected via:

$$H^1(F_{\text{multi}}) \neq 0.$$

Corrections:

$$\text{Score}(\text{correction}) = \text{Sim}(\text{corrected}, \text{context}).$$

7.3 Bio-Inspired Calibration

Shoggoths mimic cellular processes (Fant, 2007).

7.4 3D Anchoring and Error Memory

Errors are stored with coordinates for AR/VR overlays.

8 Innovative Implications

Shoggoths' capabilities enable transformative AI, leveraging RSVP, BFO, NCL, and Polycompiler/Yarncrawler metaphors for applications in neurosymbolic AI, ethical AI, quantum-inspired computing, bio-inspired systems, robotics, and AR/VR.

8.1 Neurosymbolic AI

Shoggoths combine BFO and embeddings for code synthesis.

8.2 Ethical AI Design

Cohomological regularity embeds ethical constraints.

8.3 Quantum-Inspired Computing

Sheaf sections parallel quantum superposition.

8.4 Bio-Inspired Systems

Shoggoths emulate neural networks

8.5 Anomaly Detection

Shoggoths detect outliers using RSVP entropy spikes.

8.6 Code Synthesis

Shoggoths interpolate code via Polycompilers.

8.7 Robotics

Shoggoths process sensor data–lidar, audio–with Yarncrawlers repairing noise, for navigation.

8.8 AR/VR

3D anchoring enables real-time error correction in virtual overlays.

9 Comparisons with Other Paradigms

Shoggoths contrast with traditional algorithms, Fant's model, and biological systems, enhanced by Polycompiler/Yarncrawler metaphors.

9.1 Traditional Algorithmic Models

Shoggoths prioritize concurrency

9.2 Fant's Invocation Model

Shoggoths extend NCL with RSVP and BFO

9.3 Biological Systems

Shoggoths mirror neural firing

10 Summary and Unified Formulation

This section synthesizes the Shoggoth framework, integrating RSVP, BFO, sheaf theory, NCL, photogrammetry, error correction, edge scheduling, and Polycompiler/-Yarncrawler metaphors, formalized as derived functors.

10.1 Unified Expression

$$\begin{gathered}\text{PrioritizingShoggoth} \cong \text{DerivedFunctor}\,(\\ E : \mathcal{D} \to \mathcal{V}ec, \\ \text{Sim} : \mathcal{V}ec^{\text{op}} \times \mathcal{V}ec \to \mathbb{R}, \\ \text{Trig} : \mathbb{R} \to \text{Alerts}, \\ F : \text{Open}(X) \to \mathcal{V}ec \\ )\end{gathered}$$

Concept	Formal Structure	RSVP Role
Memory Chunks	Objects in $\mathcal{D}$	Data substrates
Embedding	Functor $E : \mathcal{D} \to \mathcal{V}ec$	Vector projection
Priority	Object in $\mathcal{P}$	Semantic attractors
Relevance Match	Hom-pairing, threshold logic	Entropic trigger
Activation Logic	NCL, modal operators	Asynchronous detection
Shoggoth Memory	Sheaf F over space X	Coherence
Attention Evolution	Simplicial object	Priority updates
Relevance Rediscovery	Temporal sheaf pullbacks	Data reactivation
RSVP Interaction	Field scanning	Φ, $\mathbf{v}$ anomalies

Table 2: Correspondences of Shoggoth Components

10.2 Philosophical Implications

Shoggoths enable reflexive, ethical cognition

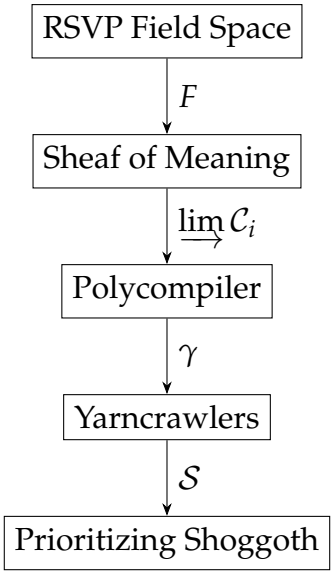

Figure 3: Shoggoth System Architecture

10.3 System Architecture

11 Future Directions

This section outlines research prospects, leveraging Shoggoths' capabilities and Polycompiler/Yarncrawler metaphors for real-time RSVP-AI, topos-theoretic generalizations, neurosymbolic calibration, and applications in anomaly detection, code synthesis, robotics, and AR/VR.

11.1 Topos-Theoretic Generalizations

Extend sheaf structures to topos theory for generalized semantics.

11.2 Hardware Implementations

Implement RSVP-AI on NCL-based chips (Fant, 2005).

11.3 Neurosymbolic Calibration

Refine BFO integration for code synthesis.

11.4 Applications

Develop demos for anomaly detection, code synthesis, robotics, and AR/VR.

12 Appendices

A Formal Definitions

- **Category** $\mathcal{D}$: Data chunks; morphisms as transformations.
- **Functor** E: $E : \mathcal{D} \to \mathcal{V}ec$.
- **Sheaf** F: Embeds open sets of X.
- **Cohomology**: $H^1(F)$ for inconsistencies.
- **NCL Threshold**: $\mathcal{N}(d) = \top \iff \mathrm{Trig}_\lambda$.

B RSVP PDE Structures

$$\frac{d\Phi}{dt} = -\alpha(\Phi - \Phi_{\text{target}}), \quad \frac{d\mathcal{S}}{dt} = \beta(\text{novelty} - \text{resolution}), \quad \partial_t \mathbf{v} + (\mathbf{v} \cdot \nabla)\mathbf{v} = -\nabla\Phi - \nabla\mathcal{S}.$$

Conscious complexity: $\int \Phi \cdot \mathcal{S}\, d^4x$. Numerical simulation uses finite difference methods on GPU clusters.

C Ontology Integration

The `RSVP-ontology.owl` defines RSVP fields and processes:

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns="http://example.org/RSVP-O#"
         xml:base="http://example.org/RSVP-O"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:owl="http://www.w3.org/2002/07/owl#"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <owl:Ontology rdf:about="http://example.org/RSVP-O"/>
  <!-- Classes -->
  <owl:Class rdf:about="http://example.org/RSVP-O#RSVP-Scalar-Field">
    <rdfs:comment>A scalar field that inheres in a spacetime region and evolves
        over time.</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://example.org/RSVP-O#RSVP-Vector-Field">
    <rdfs:comment>A vector field representing directional flow in the RSVP
        framework.</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://example.org/RSVP-O#RSVP-Entropy-Field">
    <rdfs:comment>A scalar field representing entropy density at each point in
        spacetime.</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://example.org/RSVP-O#Lamphrodyne-Coupling">
    <rdfs:comment>A process coupling vector and entropy fields resulting in
        entropic smoothing.</rdfs:comment>
  </owl:Class>
```

```
  <owl:Class rdf:about="http://example.org/RSVP-O#Entropic-Redshift">
    <rdfs:comment>A process involving entropy dissipation leading to redshift effects.</rdfs:comment>
  </owl:Class>
  <!-- Object Properties -->
  <owl:ObjectProperty rdf:about="http://purl.obolibrary.org/obo/BFO_0000052">
    <rdfs:label>inheres in</rdfs:label>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://purl.obolibrary.org/obo/BFO_0000057">
    <rdfs:label>has participant</rdfs:label>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://purl.obolibrary.org/obo/BFO_0000055">
    <rdfs:label>results in</rdfs:label>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://purl.obolibrary.org/obo/BFO_0000054">
    <rdfs:label>constrained by</rdfs:label>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://purl.obolibrary.org/obo/BFO_0000056">
    <rdfs:label>is realized by</rdfs:label>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://purl.obolibrary.org/obo/RO_0002418">
    <rdfs:label>causally upstream of</rdfs:label>
  </owl:ObjectProperty>
  <!-- Restrictions -->
  <rdf:Description rdf:about="http://example.org/RSVP-O#RSVP-Scalar-Field">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    <purl:obo/BFO_0000052 rdf:resource="http://example.org/RSVP-O#Spacetime-Region"/>
    <purl:obo/BFO_0000057 rdf:resource="http://example.org/RSVP-O#RSVP-Scalar-Field-Evolution"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://example.org/RSVP-O#Lamphrodyne-Coupling">
    <purl:obo/BFO_0000057 rdf:resource="http://example.org/RSVP-O#RSVP-Vector-Field"/>
    <purl:obo/BFO_0000057 rdf:resource="http://example.org/RSVP-O#RSVP-Entropy-Field"/>
    <purl:obo/BFO_0000055 rdf:resource="http://example.org/RSVP-O#Entropy-Gradient-Smoothing"/>
    <purl:obo/BFO_0000054 rdf:resource="http://example.org/RSVP-O#Scalar-Field-Gradient"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://example.org/RSVP-O#Entropic-Redshift">
    <purl:obo/BFO_0000057 rdf:resource="http://example.org/RSVP-O#RSVP-Scalar-Field"/>
    <purl:obo/BFO_0000056 rdf:resource="http://example.org/RSVP-O#Entropy-Dissipation"/>
    <purl:obo/RO_0002418 rdf:resource="http://example.org/RSVP-O#Photon-Energy-Decrease"/>
  </rdf:Description>
</rdf:RDF>
```

D Code Snippets

D.1 Alternating Completion Pseudocode

```
def alternate_completion(corpus, n=2, model_selector):
    C_prime = [line for i, line in enumerate(corpus) if i % n != 0]
    embeddings = embed(C_prime)
    completions = []
    for i in range(1, len(corpus), n):
        candidates = [model.predict(embeddings[i//n]) for model in
            model_selector.models]
        scores = [Sim(embeddings[i//n], c) for c in candidates]
        best_model = model_selector.select(candidates, scores)
        completions.append(best_model.predict(embeddings[i//n]))
        model_selector.update_weights(scores)
    return completions
```

D.2 Prioritizing Shoggoth with Photogrammetry and Error Correction

```
import faiss
import numpy as np
from openai import OpenAI
import colmap
import tesseract
import whisper
import time

class PrioritizingShoggoth:
    def __init__(self, embedding_model, threshold=0.85):
        self.embedding_model = embedding_model
        self.index = faiss.IndexFlatIP(1536)
        self.threshold = threshold
        self.priority_vectors = []
        self.metadata_store = {}
        self.error_ledger = []

    def set_priority(self, texts):
        self.priority_vectors = [self.embed(text) for text in texts]

    def embed(self, text):
        response = self.embedding_model.embeddings.create(input=text, model="
            text-embedding-ada-002")
        return np.array(response.data[0].embedding)

    def photogrammetry_3d(self, images):
        keypoints = colmap.detect_keypoints(images)
        sfm = colmap.structure_from_motion(keypoints)
        mesh = colmap.dense_reconstruction(sfm)
        text = tesseract.ocr(mesh.textures)
```

```
        return mesh, text

    def correct_errors(self, text, media_type, coordinates=None):
        if media_type == "audio":
            corrected = whisper.correct_asr(text)
        else:
            corrected = self.embedding_model.correct_text(text)
        self.error_ledger.append({
            "timestamp": time.time(),
            "location": coordinates,
            "source": media_type,
            "original": text,
            "corrected": corrected
        })
        return corrected

    def index_document(self, text, metadata=None, coordinates=None):
        vector = self.embed(text)
        id = len(self.metadata_store)
        self.index.add(np.array([vector]))
        self.metadata_store[id] = metadata or {'text': text, 'coordinates':
            coordinates}
        return id

    def rescan(self):
        results = []
        for pv in self.priority_vectors:
            D, I = self.index.search(np.array([pv]), k=50)
            for i, score in zip(I[0], D[0]):
                if score > self.threshold:
                    results.append((i, score, self.metadata_store.get(i)))
                    self.on_trigger(i, score)
        return results

    def on_trigger(self, item_id, score):
        print(f"Alert: Item {item_id} relevant (score: {score:.2f})")
```

D.3 RSVP Simulation

```
import numpy as np

def simulate_rsvp(corpus, alpha=0.1, beta=0.05, epsilon=0.5):
    Phi = np.ones(len(corpus)) # Scalar field
    S = np.zeros(len(corpus)) # Entropy field
    v = np.zeros((len(corpus), 2)) # Vector field
    for t in range(100):
        novelty = detect_novelty(corpus) # Placeholder
        resolution = resolve_anomalies(corpus)
        dPhi_dt = -alpha * (Phi - target_Phi(corpus))
        dS_dt = beta * (novelty - resolution)
        Phi += dPhi_dt
```

```
        S += dS_dt
        v += update_vector_field(Phi, S)
        if np.any(S > epsilon):
            focus_attention(corpus, np.argmax(S))
    return Phi, S, v
```

E Comparative Analysis with Fant's Invocation Model

Shoggoths extend Fant's model by:

- Integrating RSVP fields via OWL ontology.
- Using BFO for grounding (Arp et al., 2015).
- Employing sheaf theory (Bredon, 1997).
- Leveraging NCL for asynchrony (Fant, 2005).

References

Arp, R., Smith, B., and Spear, A. D. (2015). *Building Ontologies with Basic Formal Ontology*. MIT Press.

Bredon, G. E. (1997). *Sheaf Theory*. Springer.

Fant, K. M. (2005). *Logically Determined Design: Clockless System Design with NULL Convention Logic*. Wiley.

Fant, K. M. (2007). *Computer Science Reconsidered: The Invocation Model of Process Expression*. Wiley.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33, 6840–6851.

Lamport, L. (1978). Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7), 558–565.

Lawvere, F. W., and Schanuel, S. H. (2009). *Conceptual Mathematics: A First Introduction to Categories*. Cambridge University Press.

Mead, C. (2002). Collective electrodynamics: Quantum foundations and technology. *Proceedings of the IEEE*, 90(10), 1593–1602.