ResonantOS Hyperdimensional Computing (HDC) Integration Strategy

1. Purpose & Scope

This document captures our research on Hyperdimensional Computing (HDC), outlines key concepts, and defines an implementation roadmap for integrating HDC into ResonantOS. It is designed for both human and AI readers to understand objectives, rationale, and steps required.

2. Background

- ResonantOS: A custom GEM on Gemini and AnythingLLM, using system prompts, knowledge base documents, and Gemini 2.5 Pro as the AI model.
- Goal: Enhance cognitive capabilities, transparent reasoning, and contextual understanding via HDC.

3. What is Hyperdimensional Computing?

- Brain-inspired paradigm using high-dimensional vectors (hypervectors, e.g., 10,000 dimensions).
- Key operations:
    - Bundling (+): Combine multiple concepts into one vector.
    - Binding (⊗): Create relationships between concepts.
    - Similarity: Measure closeness of vectors for retrieval.
- Benefits:
    - Noise tolerance
    - One-shot/few-shot learning
    - Interpretable algebraic operations
    - Distributed, holographic memory

4. Relevance to ResonantOS

- Semantic Bridge: HDC provides compositional embeddings bridging natural language (Oracle) and logic (Logician).
- Memory Systems: Associative memory for Constitutional Shield and Five Pillars (Meaning, Intent, Awareness).

- Resonance Amplification: Quasi-orthogonal hypervectors model resonance when concepts align.
- Transparent Reasoning: Algebraic operations are verifiable.

5. Implementation Libraries

1. TorchHD (PyPI: `torch-hd`, GitHub: https://github.com/hyperdimensional-computing/torchhd)
    - Built on PyTorch, GPU/MPS acceleration
    - Production-ready examples & docs: https://torchhd.readthedocs.io
2. hdlib (PyPI: `hdlib`, GitHub: https://github.com/cumbof/hdlib)
    - Pure Python, educational focus

6. Hardware Compatibility

- Mac Mini M4 (32GB RAM):
    - Unified memory bandwidth: 120 GB/s
    - M4 Neural Engine & GPU via Metal Performance Shaders (MPS)
    - Ideal for vector operations and PyTorch MPS acceleration

7. Roadmap & Timeline

- Phase 1: Proof of Concept (1 week)
    - Install TorchHD, run basic tutorials
    - Encode core concepts as hypervectors
    - Test simple binding/bundling queries
- Phase 2: Core Integration (2–3 weeks)
    - Integrate HDC memory layer into ResonantOS Custom GEM
    - Replace embedding retrieval with HDC similarity search
    - Prototype Semantic Bridge with HDC
- Phase 3: Advanced Features (1–2 months)
    - Multi-agent coordination via hypervector composition
    - Real-time reasoning logs and audit trails
    - Performance tuning on M4 hardware

8. Example Code Snippet

```
python

import torchhd
```

```python
import torch

# Dimensions
d = 10000
# Generate hypervectors
keys = torchhd.functional.random_hv(3, d)
values = torchhd.functional.random_hv(3, d)

# Bundling and binding
memory = keys * values + keys * values
# Query similarity
score = torchhd.functional.cosine_similarity(keys, memory)
print(score)
```

9. Useful Links

- HDC Overview (Nature): https://www.nature.com/research-intelligence/nri-topic-summaries/hyperdimensional-computing-and-its-applications-micro-307301
- Wikipedia: https://en.wikipedia.org/wiki/Hyperdimensional_computing
- TorchHD Docs: https://torchhd.readthedocs.io
- hdlib GitHub: https://github.com/cumbof/hdlib
- Vector Symbolic Architectures Survey: https://arxiv.org/pdf/2111.06077.pdf

10. Next Steps

1. Allocate development time in schedule.
2. Recruit or assign a developer familiar with Python and PyTorch.
3. Begin Phase 1 PoC and document findings.
4. Iterate based on results and refine integration.

Prepared by Manolo Remiddi, September 27, 2025