Efficient Memorization Strategies for Object Recognition with a Multi-Core Architecture

George Viamontes, gviamont@atl.lmco.com; Mohammed Amduka, mamduka@atl.lmco.com;
Jon Russo, jrusso@atl.lmco.com; Matthew Craven, mcraven@atl.lmco.com;
ThanhVu Nguyen, tnguyen@atl.lmco.com

Lockheed Martin Advanced Technology Laboratories

Introduction

Architectural research is increasingly driven by application needs in different domains. Research and development efforts must systematically identify the prominent properties, especially the amount of inherent parallelism and memory access/structural patterns for a wide range of emerging applications. One such application domain is recognition. Machine vision, object/scene understanding, and natural language processing belong to this domain. Recognition systems solve what is called the inverse problem. Whether it is recognizing a target from synthetic aperture radar (SAR) or image data, recognizing speech, understanding human language, identifying a person, or tracking a computer attack, the problem centers around recognizing an event, action, or object of interest in the presence of uncertainty. Highly data-parallel preprocessing, along with multi-granularity task and data-parallel learning and inference, are characteristic of machine recognition systems.

Ideally, a recognition system will perform library matching or classification based on features extracted from a model generated by an appropriate sensor. Regardless of the particular recognition application or the complexity of the sensor, recognition systems typically perform some type of data preprocessing, feature extraction, and matching or classification. While the first two stages can be computationally intensive, the final stage of matching or classification is memory access intensive. As the gap between memory and processor speed grows, by Little's Law the amount of concurrency needed to hide the latency of memory accesses will continue to increase. Memory access therefore quickly becomes the bottleneck in a multi-core implementation of the inference mechanism for matching or classification.

To attack this problem, designers must focus on innovative implementations of multi-core systems, such as embedding memory alongside the inference logic. In this work, we demonstrate such an implementation for a geometric Bayesian inference algorithm applied to object recognition. Our particular multi-core mapping accounts for the memory bottleneck and adapts a memorization-based architecture to implement the inference logic. We simulate this implementation on the ISIS framework and demonstrate its effectiveness as compared to a multi-core implementation with a monolithic memory structure. Our simulation results also indicate optimal sizes for the embedded memory for various object recognition benchmarks.

This work was sponsored by DARPA/IPTO in the Architectures for Cognitive Information Processing (ACIP) program;
Contract number FA8750-04-C-0266
Not For Public Release

Geometric Bayesian Inference

We map an object recognition algorithm hybrid that combines Bayesian inference with the popular geometric hashing method [1] to our proposed multi-core architecture. Geometric hashing is well-known in the computer-vision and medical imaging-research areas for its low-complexity and accuracy. The addition of heuristic Bayesian techniques, similar to those described in [2], makes the algorithm more tolerant of fuzzy objects such as distorted images or rotational and translational variants of the same image.

The main idea in this algorithm is to compute a hash function of salient features of the object, which maps to a location in an artificial "hash space" in which simple distance metrics can be computed between the hashed location and nearby entries containing library features to match against. As a result, our architecture must implement a form of approximate hashing that returns multiple entries within a cutoff distance of the hash location, and each entry is weighted by its distance. Models in our library whose features receive the highest sum of weighted hits are considered matches.

Architectural Mapping

One obvious choice for implementing geometric hashing is the Content Addressable Random Access Memory (CA-RAM) architecture [3], which places a hash function logic alongside conventional memory structures. Although this architecture can, in principle, be extended to perform the approximate hybrid approach, a natural extension to CA-RAM that incorporates approximate matching is the Programmable Object Evaluation Memory (POEM) architecture [4].

The POEM architecture (Figure 1) matches stored content to an input feature vector (operand) through the user-definable distance function "F". Our first implementation of POEM on a Xilinx Virtex-2 has four, dual-port banks of memory, each 256 bits wide by 512 words deep. Each 256-bit row of memory represents eight integers of stored feature content. The four banks of memory are compared to new input feature vectors in parallel, according to the

specified function, and the results are fed into a pipelined minimization tree. The data object associated with the best-responding row is returned.

The Traveling Salesman Problem (TSP) was chosen to test the recall ability and accuracy of such a memory. In learning mode, candidate problems are solved optimally, and the problem/solution pairs are stored. A replacement policy, which favors replacing least frequently used content with new inadequately represented content, flattens the response error.

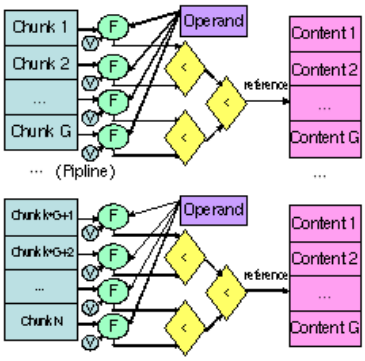

Figure 1: POEM architecture.

In this application, "F" implements the hash distance, and the subsequent POEM logic performs the weight summations.

To effectively scale a multi-core POEM architecture to the geometric inference problem, we must ensure that the match logic does not idle due to data starvation. This unwanted behavior often occurs in a multi-core architecture with a monolithic memory structure in which the bottleneck quickly becomes the communication between memory and the cores (Figure 2). One way to address this issue is to split the memory into chunks, where each chunk is dedicated to a particular core. Since each core and memory chunk is independent in this scheme, having a systematic way of assigning each core to a separate piece of the problem is important.

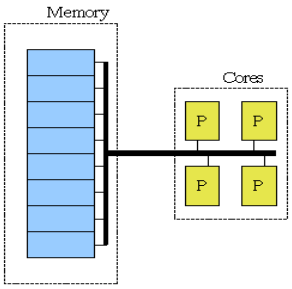

Figure 2: A multi-core architecture with a monolithic main memory.

The way chunking memory is done and the effectiveness of the approach varies with the problem. In the case of Bayesian inference for object recognition, the problem can be partitioned based on feature location within the image. In our approach, we recursively divide up the relevant features of images in the library into quadrants (Figure 3). The smallest regions are addressed via row and column bits, where each bit represents recursion into a smaller set of quadrants. Each of the smallest regions are assigned to a core and memory chunk. When matching occurs, features from the image under analysis are dispatched to the appropriate core and memory chunk based on the location of the feature. This dispatching process is efficient and requires little logic since it is based on nothing more than a grid discretization of the image.

A key parameter that must be adjusted to optimize performance in our approach is the memory-chunk size. This parameter depends not only on the location density of image features, but also on the available number of cores and the desired granularity. In other words, more cores can be used to either match against larger image regions in larger images (potentially requiring larger memory chunks) or to match against smaller regions in smaller images (an increase in the granularity, which potentially requires smaller memory chunks). We explore this critical tradeoff via simulation in the ISIS framework [5].

ISIS contains a set of C++ classes that describe the functionality of various components of a parallel computer, such as processors, memories, buses, and routers. These components can be assembled into a simulator and are easily extended to support new features. ISIS supports Message Passing Interface (MPI) and cache coherence protocols for communication between processor elements. With ISIS, we compare a monolithic-memory, multi-core architecture to our proposed architecture on various object recognition benchmarks and provide a variety of statistics including cycle counts, memory access patterns, and communication latency. The results indicate that our proposed architecture scales much better as the number of cores is increased.

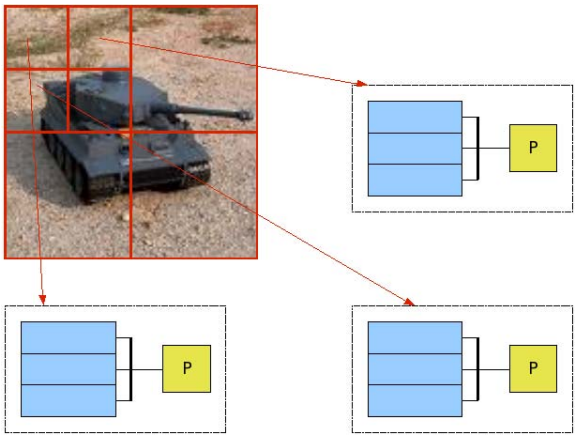

Figure 3: The proposed architecture for Bayesian inference object recognition. The image is partitioned recursively into quadrants, and each region (addressable by row and column bits) is assigned to a core and memory chunk.

References

[1] H. J. Wolfson and I. Rigoutsos, "Geometric Hashing: An Overview," *IEEE Computational Science and Engineering,*

This work was sponsored by DARPA/IPTO in the Architectures for Cognitive Information Processing (ACIP) program;
Contract number FA8750-04-C-0266
Not For Public Release

Vol 4, No. 4, pp. 10-21, 1997.

[2] I. Rigoutsos and R. Hummel, “A Bayesian Approach to Model Matching with Geometric Hashing,” *Computer Vision and Image Understanding*, Vol. 62, No. 1, pp. 11-26, 1995.

[3] S. Cho, J. R. Martin, R. Xu, M. H. Hammoud, and R. Melhem, “CA-RAM: A High-Performance Memory Substrate for Search-Intensive Applications,” *In Proc. of the IEEE Intl. Symposium on Performance Analysis of Systems and Software (ISPASS 2007)*, pp. 230-241, 2007.

[4] J. C. Russo, M. Amduka, K. Pederson, R. Lethin, J. Springer, R. Manohar, and R. Melhem, “Enabling Cognitive Architectures for UAV Mission Planning,” *In Proc. of the Tenth Annual High Performance Embedded Computing Workshop (HPEC 2006)*, 2006.

[5] T. Horita and M. Wakabayashi, “Environment for Multiprocessor Simulator Development,” *In Proc. of the Intl. Symposium on Parallel Architectures, Algorithms and Networks (ISPAN 2000)*, p. 64, 2000.

This work was sponsored by DARPA/IPTO in the Architectures for Cognitive Information Processing (ACIP) program;
Contract number FA8750-04-C-0266
Not For Public Release