Pattern Recognition without Tradeoffs: Scalable Accuracy with No Impact on Speed

Rick Dove
Kennen Technologies and Stevens Institute of Technology
rick.dove@kennentech.com

Abstract

Automated recognition of patterns in data is constrained by tradeoffs among speed, cost, and accuracy. A new reconfigurable VLSI processor architecture decouples the speed/accuracy tradeoff, and renders the cost/accuracy tradeoff negligible, enabling new performance and new applications. The architecture features massively-parallel, dynamically-configurable finite-state-machines which simultaneously process the same data stream. Low cost VLSI fabrication, unbounded scalability, and high-speed constant-rate throughput independent of pattern number and complexity breaks current trade space constraints. This paper introduces features of the processor architecture responsible for the decoupling, and shows how current tradeoff structure is altered.

1. Introduction

Design tradeoffs occur when two or more performance metrics are locked in an inverse relationship. To get more of one means less of another. In pattern recognition we attempt to find some sense in raw data, but not if it takes so long we can't act on the sense making. Conversely, we want knowledge in time to use it, but not if it's false knowledge.

Progress in pattern recognition has come in the form of trying harder: more elaborate recognition algorithms, pattern-tuned special purpose FPGA processors, multi-core parallelism, clustered servers, and multiple graphic processors. All of these approaches continue to make tradeoffs among the same forces in tension: accuracy, time, and cost. Time and cost are generally fixed limit requirements, with accuracy the compromised variable.

Biological capability is the benchmark for pattern recognition. People considered truly expert in a domain (e.g. chess masters, medical diagnosticians) are thought unable to achieve that level until they've accumulated some 200,000 to a million meaningful patterns [1]. The accuracy of their sense making is a function of the breadth and depth of their pattern catalog. Interestingly, in biological entities, the accumulation of large expert-level pattern quantities does not manifest as slower recognition time. The capabilities of biological pattern recognizers appears to argue for massive pattern libraries rather than heuristics or reasoned inference.

Constrained by the nature of computing mechanisms and recognition algorithms employed in this service, automated systems execute time-consuming sequential steps to sort through possible patterns, often performing statistical feature mathematics as well.

Figure 1 depicts the general relationship between classification time and the number of classification patterns in the sequential search approach employed for packet-header classification by Snort 2.6 [2]. Data and depiction comes from a 2008 published experiment [3] on this widely employed Network Intrusion Detection System. The purpose here is strictly to demonstrate the shape of the relationship when sequential search is the approach, as other

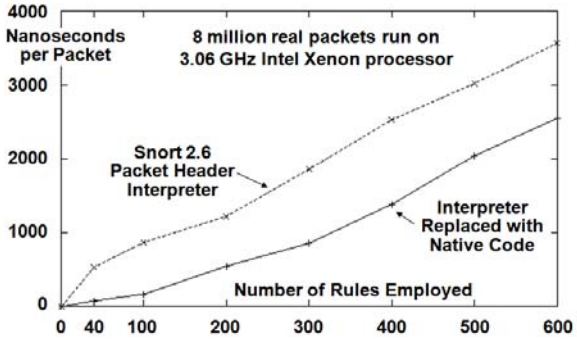

Figure 1: Experimental data borrowed with permission from [3] showing dependence of pattern classification time on number of rules (patterns) employed for intrusion detection packet-header classification. The authors compare the standard Snort 2.6 interpreted approach to their native-code approach. In both cases, the average time increases as the number of possible classifications increases.

approaches are possible. Snort 2.6, for instance,

employs an Aho-Corasic [4a] parallel approach for its second phase of pattern detection, that of packet content detection.

Stored reference patterns have some form of representation, whether this is a string of syntactically structured symbols, like the packet headers of Figure 1, or a vector of features each with statistical gradients, perhaps representing symptoms that indicate a diagnosis. The time taken to classify data as one of these known reference patterns is a function of both the quantity of patterns and the complexity of each of those reference patterns. In some applications, like genetic biosequence alignments where exact matches are unlikely, every stored reference pattern must be evaluated in order to find which is most similar to the data being presented. Faster processors, more processors, and more clever algorithms don't change the fact that speed and accuracy are both functions of reference pattern quantity and complexity.

Breaking the linkages between the tradeoff variables would change the nature of the game. How a new general-purpose pattern processor does that will be shown in this paper.

This paper describes those aspects of a very large scale integrated (VLSI) processor architecture that severs the linkages between the time it takes to detect (classify) a pattern and the number and complexity of the patterns that are detectable, at a cost potentially well below popular general purpose processors.

Parallel recognition-algorithm development is facilitated by an emulator and reference-pattern compiler [5a], and by a Software Developer's Kit (SDK) [6a]. FPGA prototype boards are also available in advance of VLSI hardware availability. Throughout this paper we refer to the VLSI instantiation of the architecture as the "processor".

Section 2 reviews the tradeoff problem as expressed in the literature. Section 3 presents and discusses the processor architecture as it relates to pattern capacity and classification time. Section 4 discusses the way that tradeoffs are decoupled. Section 5 provides final remarks on implications and opportunities.

2. The Tradeoff Problem

In sequential classification, more accurate recognition comes at the expense of speed and/or cost. To get necessary speed, a system reduces the quantity and complexity of reference patterns, increasing error. To get necessary accuracy, a system increases quantity and complexity of reference patterns, increasing time and/or cost.

Addressing the issues of memory resources, especially for large numbers of patterns, [7a] notes the common practice of compressing the number and quality of patterns, as well as the incoming sensory data – resulting in an inherent tradeoff between data representation expense and the complexity of reliable patterns. The result in automated systems is noted by [8a] as making not-fully-reliable inferences from compressed data.

To put this in the context of Network Intrusion Detection Systems (NIDS), open-source Snort [2] is the most-widely-used library of intrusion signatures, with a current catalog of almost 9,000 signatures at an average size of about eight characters. Some voices are saying that signature-based IDS is obsolete, with behavior-based systems taking over. Though network security administrators recognize considerable false positive and false negative issues with traditional signature based systems, a decline in usage is not yet evident. Apparently behavior based systems are augmenting, rather than replacing, signature based systems. It is likely that signature based systems will always have a place as fast filters for known problems.

With an average signature size (for Snort) less than ten characters, and a catalog of less than 9,000 signatures, it is little wonder that accuracy is wanting. Removing system imposed limits on pattern quantity and complexity of signatures could change the nature of "signatures" and open up new possibilities. One can observe that behavior detection is signature detection at a different level, simply with different features.

Summing up the accuracy impact, [7a] relates classification error to compressed pattern representations, done to speed up classification time and, in some cases, reduce memory costs:

> "Feature extractors are typically designed with the objectives of transforming the raw data available to the system into a format which facilitates easy matching or storage, and is robust ("invariant") with respect to characteristic signature variations in sensory data. ... Whatever the motivations are, the crucial common aspect of all data encoding operations for our present purposes is that they reduce the amount of data available to the system as compared with the original data (usually in a lossy manner)."

In sum, tradeoffs rule and limit, most especially in cost conscious commercial applications.

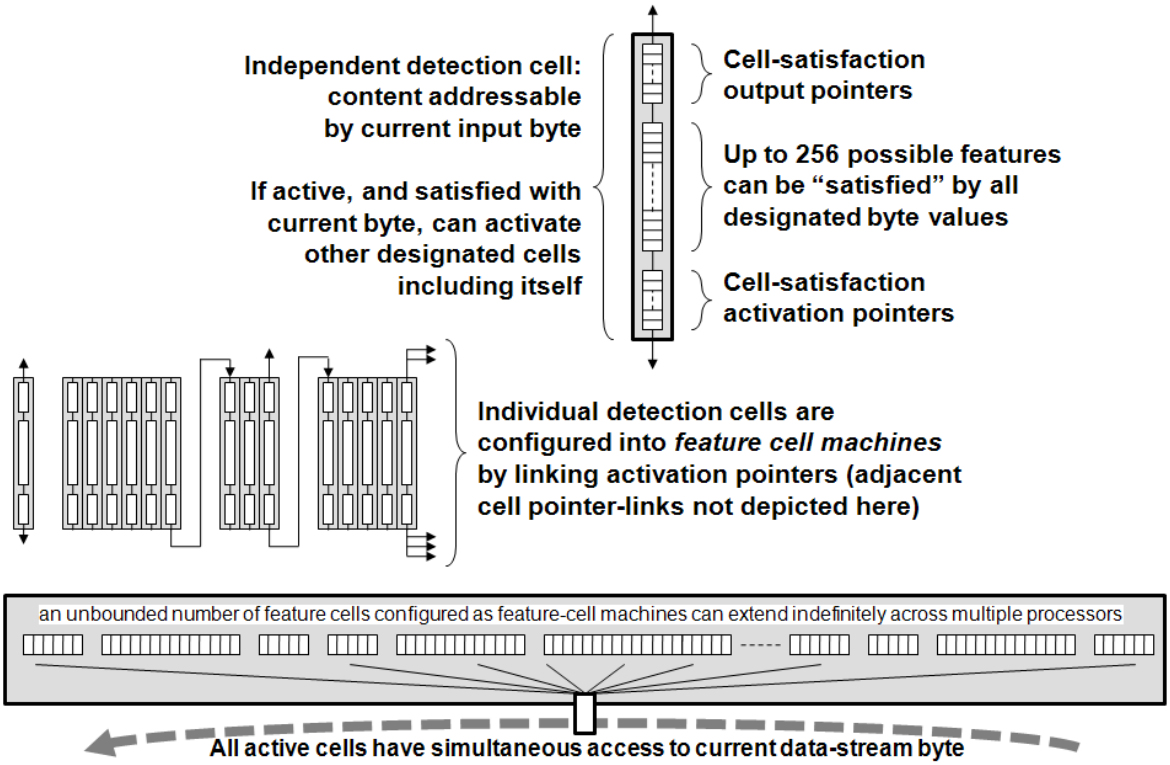

Figure 2: Configurable feature cells and Feature Cell Machines (FCMs).

3. Processor Architecture

Referring to Figure 2, a partial view of the processor's architectural concept shows massively replicated detection cells. A half million such cells on a single VLSI chip appears possible for early generation silicon. These cells are independent units, with four dynamically configurable elements to consider here: 1) an activation status, 2) a 256 element feature-vector designating all byte values of interest, 3) a set of pointers to other cells that will be activated if this cell is "satisfied", and 4) a set of pointers to output transforms that can logically combine designated cell-satisfactions into buffered output and reset designated processor status.

In operation, an external controller feeds data stream bytes in sequence to a current-byte register in the processor. The presentation of each new byte triggers the beginning of a detection cycle. The current byte acts as an index into the feature vector for all active cells simultaneously. If a cell is active and the current byte value is one of interest, as designated in the feature vector, it is said to be "satisfied". That satisfaction will activate (for next cycle) all other cells according to the satisfied cell's activation pointers, and will cause designated output transformations to occur according to this cell's output pointers. A cell's activation pointers may include one that reactivates itself, as cell activation is effective for a single cycle only. Note that a cell can respond to any number of data-stream byte values – which enables value-based as well as syntactic feature-based classification.

Important to note, multiple processors can be employed in parallel and serial arrangements to increase throughput speed and/or reference-pattern capacity. Interleaving packet-based data streams, for instance, across multiple processors can be used to increase throughput speed. Presenting the data stream "current" character to multiple processors simultaneously can be used for unbounded reference-pattern scalability.

With this architecture, groups of detection cells can be configured into Feature Cell Machines (FCMs), similar to finite state machines, by setting activation pointers to pass activation successively through a group of successively "satisfied" cells. One cell may activate many other cells, so that multiple pattern branches may become simultaneously active. Any number of such FCMs may be configured within the total cells available within a processor. Typically such

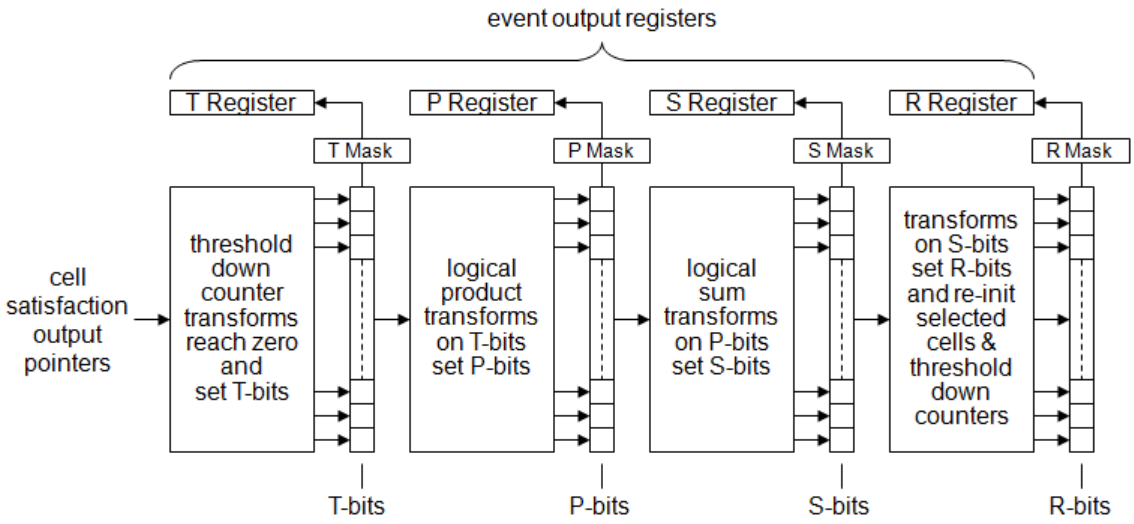

Figure 3: Configurable output Transform Machines (TMs).

FCMs are created to detect (classify) specific patterns or sub-patterns of interest.

In addition to Feature Cell Machines, the processor provides configurable Transformation Machines (TMs). Shown in Figure 3, these TMs are the link between satisfied feature cells and processor output. TMs transform multiple feature cell satisfaction events into bit settings in four types of output registers: threshold, logical product, logical sum, and re-initialization. Masks are used to gate which bit values are placed in event output registers. The values of down counters may also be read as output at any time by an external controller. Cell satisfaction output pointers may be configured as latched or unlatched: latched can only cause output once, unlatched will cause output every time the cell is satisfied.

Figure 3 provides sufficient detail for the purpose here, and no more will be said of Transform Machines at this time. Each TM may be configured to generate multiple-byte output from the processor immediately during the same cycle causing cell satisfaction.

Processor cycles are driven by the data stream input. Each byte of the data stream initiates a cycle as it is fed to the processor by an external controller. An upper limit exists on the rate at which data stream bytes will be accepted by the processor that is dependent upon specifications particular to the processor model – with 1 gigabit rates possible in early generation silicon. Latency from time of data byte presentation to register output caused by a particular byte satisfaction event is fixed and on the order of 2 to 3 cycles.

One last architectural concept is worth noting: the features cells and transformations machines are segmented into multiple blocks within a processor – so that multiple groups of TMs may be active simultaneously. Output is buffered so that multiple TM groups will not interfere with each other.

Some interesting capabilities are worth noting. For one, a single detection cell can be satisfied by multiple byte values, e.g., the ten numerical characters, or the three sentence-end characters, or both upper and lower case of a specific character, or even all of the 256 possibilities for a don't-care situation. For another, a satisfied cell that reactivates itself as well as at least one other cell can be used as a wildcard (*) function to skip over any number of data stream bytes of no import. This same self-reactivation capability can be used to cause multiple traces to be active simultaneously in one or more FCMs. Exploring the uses and values of these capabilities in parallel recognition algorithms is beyond the scope of this paper, and is taken up elsewhere [9a].

Cells can define what are referred to as *primitives* and simple *features* in the pattern recognition literature. FCMs can be configured from cells to detect an ordered set of primitives as a sub-pattern, or as a complete but simple pattern, typical of common network intrusion detection signatures.

From the discussion it is evident that data stream throughput rate is neither affected by the size (number of primitives) within a subpattern nor the quantity of subpatterns. Should more detection cells be needed than a single processor can provide, processors configured for different reference patterns can work on the same data stream simultaneously, offering unbounded scalability without impact on throughput speed.

4. Tradeoff Decoupling

Accuracy, speed and financial cost are generally the first order tradeoffs in pattern recognition systems.

4.1 Speed

The processor is quasi systolic, synchronized by the arrival of a data stream byte, with a maximum fixed latency (to results output) of 2.5 input bytes at maximum presentation speed.

An external I/O controller may present data stream bytes at any rate up to the maximum that the processor can accommodate (design simulations indicate first generation silicon will accommodate a minimum of 1 Gb/sec.). Both latency and throughput are thus a function of data-stream presentation rate, and completely independent of how many cells may be active simultaneously.

4.2 Capacity

Simulations indicate first generation silicon can supply about 512k detection cells per processor. These detection cells, equivalent to pattern primitives, can be configured into any number of reference-pattern finite state machines. Any number of detection cells may be active simultaneously, meaning that the number of reference patterns, either active or passive at any moment, has no bearing on data-stream throughput. When more than one-processor capacity is needed, the architecture scales indefinitely across multiple processors, with no impact on throughput speed. There are no relationships between reference pattern capacity and processing speed, and thus no speed penalty for utilizing large numbers of lengthy highly-discriminating reference patterns.

4.3 Complexity

As the discussion of processor architecture pointed out, pattern complexity comes into play a few ways. At the low level, a single detection cell provides a complex pattern primitive compared to traditional string-matching resources: any number of 256 byte values can provide satisfaction, a cell can loop on itself for the wildcard function, a cell can activate any number of other cells, and a cell can provide the don't-care function. At the high level, counter and logical expression Transformation Machines [9a] capabilities provide rich pattern aggregation from subpatterns – all within the fixed cycle time

4.4 Cost

Market price will be set by the VLSI producer, with production cost the most important factor. Neither market price nor production cost can be projected here in absolute terms so we will contrast production cost factors.

A dominant architectural feature enables low production cost relative to typical processor chips: massive replication of simple, standard-cell circuitry.

In contrast, arithmetic instruction-based processors, commonly employed for pattern recognition, such as multi-core processors and GPUs (Graphics Processing Units), involve a large amount of highly specific logic circuitry to perform many different instruction-specific functions (e. g., floating point multiplication, and division) and sophisticated work flow management (e. g., instruction pipelining and caching).

Design cost of the complex vs. the simple is dramatically different, but probably not as cost-affecting as production yield – the percentage of the integrated circuits on a wafer that pass test and get packaged as salable product. The architecture lends itself to redundant cell substitution during testing should faulty circuits be found, a mature practice for some time in both memory and PLD (programmable logic device) production.

The result should be massive quantities of pattern capacity available for relatively inconsequential costs. This eliminates (relatively speaking) the cost motivation to skimp on reference pattern quantity and complexity. Of course breaking through this barrier by today's standards will encourage new applications impractical in the past, and eventually a new cost tradeoff point will emerge for massive reference pattern applications.

4.5 Accuracy

Accuracy is the inverse of error. Error is a function of reference pattern complexity and reference pattern length (assuming reference pattern appropriateness). Longer patterns and more patterns increase accuracy. With no speed penalty for more patterns or longer patterns, and negligible cost effect, accuracy is not bound by these traditional first order tradeoffs.

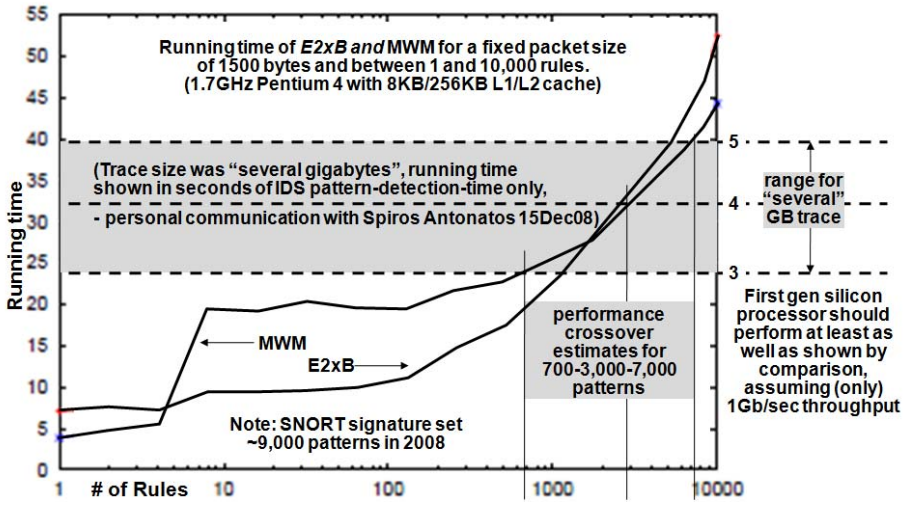

Figure 4: Experimental data reprinted with permission from [10a]. Added overlay shows crossover range for a 1 Gb/sec pattern processor.

4.6 Some comparison

In [10a] there is a performance analysis of content matching NIDS that, among other criteria, measures the effect of the number of rules (patterns) on classification time for two different high-performance classification algorithms. These algorithms weave all patterns into a single tree that has the effect of a parallel search of all patterns simultaneously. These algorithms are variants of the well-known Aho-Corasic [4a] model for this parallel approach. The authors observe that the running time increases super linearly above 2,000 rules.

Figure 4 is borrowed with permission from [10a] with an overlay of additional information. Here we see the processor has a flat-line crossover characteristic that might intersect the high performance algorithms somewhere around 3,000 rules. The intersection is an approximation based on a personal communication with the lead author's estimate of "several gigabytes" of processed traffic. Of course a faster Intel (or other) processing unit would surely turn in faster times, though larger pattern capacity does cause more memory access when L2 cache is exceeded. The purpose of the comparison here is not to peg a static crossover point, but rather to show the nature of flat-line crossover as a function of pattern quantity.

Referring back to Figure 1, where experimental data shows the time it takes to classify headers for Snort pattern groupings, the crossover point is more dramatic. Header classification requires 24 bytes of processing. At a conservative 1 Gigabit per second this takes 190 nanoseconds. This would be approximately 20 rules of interpreted headers or 150 rules for native-code processing.

5. Final Remarks

In summary, the discussions of architecture and tradeoff decoupling show that processing time is not a function of reference pattern size, reference pattern quantity, or reference pattern complexity. Processing time is only a function of the quantity of data to be processed and the maximum presentation speed allowed by the processor.

The processing architecture described here enables inexpensive and unbounded patterns quantities, with no effect on processing speed. This brings new options to existing recognition tasks and enables new applications impractical previously. It should offer opportunities to decrease false positives and false negatives simultaneously in current NIDS applications simply with more signatures and more signature complexity.

Ignoring special-purpose and application-tuned hardware processors, general pattern recognition algorithms assume a sequential-instruction, stored-program processor is employed as a key or sole processing resource of the classification engine. Multi-core processors, server clusters, and grid computing break the problem into pieces that can be done in parallel, but sequential-instruction processing engines are still employed.

This new processor's massively parallel architecture

warrants new ways of thinking about the pattern recognition problems in all domains.

In 2009 we began work on a challenge-wiki for developing parallel recognition Algorithms [11a]. This is a wiki project separated into application domains, with initial work started on demonstration algorithms for aspects of data fusion, aberrant behavior detection in autonomous-agent systems, data mining, and intrusion detection. More topic areas will be added as wiki contributors multiply and feel inclined. The work is supported with a downloadable emulator for algorithm proof-of-concept demo development.

In conclusion, this processor's capabilities are likely to encourage new applications with extremely large pattern bases. Take the human-expertise estimates we opened with, for instance, of 200,000 to one million stored patterns. For argument, we might assume that such patterns average seven features (the accepted rough human limit of immediate comprehension), with seven possible values each (in a specific domain of expertise). That would permit an expert's experience catalog of 823,543 meaningful reference patterns, consistent with the estimates. Probably just a coincidental number, but at least now in the realm of actionable exploration.

Acknowledgment

This work was supported in part by the U.S. Department of Homeland Security award NBCHC070016. Curt Harris invented the processor architecture, Jim Burkhard developed the SDK and Emulator, and with Jack Ring and Tammy Wenderlich, all contributed to the understandings in this paper

References

[1] Philip Ross, "Flash of Genius," an interview with Herbert Simon, *Forbes*, November 16, 1998, pp. 98-104.

[2] Martin Roesch, Snort - Lightweight Intrusion Detection for Networks, 13th Systems Administration Conference, USENIX, 1999, www.snort.org.

[3] Alok Tongaonkar, Sreenaath Vasudevan, R. Sekar, Fast Packet Classification for Snort by Native Compilation of Rules, Proceedings of the 22nd Large Installation System Administration Conference (LISA '08), USENIX, Nov 9–14, 2008.

[4] A. Aho, M. Corasick, Fast pattern matching: an aid to bibliographic search, Communication of the ACM, 18(6):333-340, June 1975.

[5] Kennen Technologies, USee Emulator User's Guide, Version 7.5, August 2008, inquire rick.dove@kenentech.com.

[6] Kennen Technologies, *Software Developer's Kit User's Guide*, Version 2.2, July 2008, inquire rick.dove@kenentech.com.

[7] M. Brandon Westover and Joseph A. O'Sullivan, "Achievable rates for pattern recognition," arXiv:cs/0509022v1 [cs.IT], September 2005, Available: http://arxiv.org/PS_cache/cs/pdf/0509/0509022v1.pdf.

[8] Hongche Liu, Tsai-Hong Hong, Martin Herman, and Rama Chellappa (1996), "Accuracy vs. Efficiency Trade-offs in Optical Flow Algorithms," *Computer Vision and Image Understanding*, Academic Press, pp 271-286.

[9] Kennen Technologies, *USee ™ Algorithm Developer's Guide for Concurrent Patterns*, Kennen Technologies Technical Report, inquire rick.dove@kennentech.com.

[10] S. Antonatos, K. G. Anagnostakis, E. P. Markatos, M. Polychronakis, Performance Analysis of Content Matching Intrusion Detection Systems, International Symposium on Applications and the Internet, 2004

[11] Kennen Technologies, Parallel Pattern Algorithm Challenge Wiki, inquire rick.dove@kennentech.com.8a