INTENSIVE 2012

The Fourth International Conference on Resource Intensive Applications and Services

ISBN: 978-1-61208-188-5

March 25-20, 2012

St. Maarten, Netherlands Antilles

INTENSIVE 2012 Editors

Pascal Lorenz, University of Haute Alsace, France

Petre Dini, Concordia University, Canada / China Space Agency Center-Beijing, China

INTENSIVE 2012

Foreword

The Fourth International Conference on Resource Intensive Applications and Services (INTENSIVE 2012), held between March 25-30, 2012 - St. Maarten, Netherlands Antilles, addressed a large spectrum of topics related to technologies, hardware, software and mechanisms supporting intensive applications and services (IAS).

Intensiveness is a qualitative metrics expressing the degree of resources needed to fulfill a given task under strong requirements of either communication, computation, understanding, storage, data-volume, or collaboration, where solutions are time-critical or have a mass impact. The well-known computation/resource intensive paradigm portrays a paradigm shift with the advent of high-speed applications, on-line multi-user game services, GRID applications and services, or on-demand resources and services. With the heavy distributed and parallel applications, communication intensive aspects, such as bandwidth-intensive, multicast-intensive, and propagation intensive, became key contributors for optimizing workflows of computation of intensive tasks, or storage and access-intensive databases. For example, the massive scalability and storage capacity make it the clear choice for replication-intensive applications; the bandwidth-intensive becomes relevant for content streaming systems, while replication and data reuse are important for data-intensive applications on GRIDS. Data-intensive computing is another view on intensiveness, where data availability and data volume may impact solutions for time-critical aspects.

We welcomed technical papers presenting research and practical results, position papers addressing the pros and cons of specific proposals, such as those being discussed in the standard forums or in industry consortia, survey papers addressing the key problems and solutions on any of the above topics short papers on work in progress, and panel proposals.

We take here the opportunity to warmly thank all the members of the INTENSIVE 2012 technical program committee as well as the numerous reviewers. The creation of such a broad and high quality conference program would not have been possible without their involvement. We also kindly thank all the authors that dedicated much of their time and efforts to contribute to INTENSIVE 2012. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

We hope that INTENSIVE 2012 was a successful international forum for the exchange of ideas and results between academia and industry and to promote further progress in resource intensive applications and services

We are certain that the participants found the event useful and communications very open. The beautiful places of St. Maarten surely provided a pleasant environment during the conference and we hope you had a chance to visit the surroundings.

INTESIVE 2012 Chairs
Pascal Lorenz, University of Haute Alsace, France
Petre Dini, Concordia University - Montreal, Canada / China Space Agency Center - Beijing, China

INTENSIVE 2012

Committee

INTENSIVE 2012 Technical Program Committee

Scott Alexander, Telcordia Technologies, Inc. - Piscataway, USA
Giner Alor Hernandez, Instituto Tecnologico de Orizaba - Veracruz, México
Budak Arpinar, University of Georgia, USA
Rudolf Berrendorf, Bonn-Rhein-Sieg University of Applied Sciences - Sankt Augustin, Germany
Jonathan Blackledge, Dublin Institute of Technology, Ireland
Fernando Boronat Seguí, Universidad Politécnica de Valencia, Spain
Gerardo Canfora, University of Sannio - Benevento, Italy
Li-Der Chou, National Central University - Taipei, Taiwan, ROC
Nicholas John Dingle, Imperial College London, UK
Juan Carlos Dueñas López, Universidad Politécnica de Madrid, Spain
Hans-Dieter Ehrich, Technische Universität Braunschweig, Germany
Paul Gibson, TELECOM SudParis, France
Vic Grout, Glyndwr University - Wrexham, UK
Jameleddine Hassine, King Fahd University of Petroleum & Minerals (KFUPM), Saudi Arabia
Wolfgang Hommel, Leibniz-Rechenzentrum - Garching, Germany
Paul Humphreys, University of Ulster, UK
Chih-Cheng Hung, Southern Polytechnic State University, USA
Félix J. García Clemente, Universidad de Murcia, Spain
Jon Hall, Open University, UK
Robert J. Hilderman, University of Regina, Canada
Jinlei Jiang, Tsinghua University - Beijing, China
Scott A. Klasky, ORNL, USA
Evangelos Kranakis, Carleton University - Ottawa, Canada
Mario Marcelo Berón, National University of San Luis (Argentina) / University of Minho, Portugal
Eda Marchetti, ISTI-CNR - Pisa, Italy
Michael J. May, Kinneret College on the Sea of Galilee, Israel
Shawn McKee, University of Michigan, USA
René Meier, Trinity College Dublin, Ireland
Carlos Julian Menezes Araújo, Federal University of Pernambuco, Brazil
Jose Merseguer, Universidad de Zaragoza, Spain
Marcellin Julius Nkenlifack, University of Dschang - Bandjoun, Cameroun
John Paul O'Neill, Trinity College Dublin, Ireland
Meikel Poess, Oracle Corporation, USA
Miodrag Potkonjak, University of California - Los Angeles, USA
Sean Rooney, IBM Research - Zurich, Switzerland
Arun Saha, Fujitsu Network Communications, USA
Rainer Schmidt, AIT Austrian Institute of Technology GmbH, Austria
Javier Soriano, Universidad Politécnica de Madrid, Spain
George Spanoudakis, City University London, UK
Parimala Thulasiraman, University of Manitoba, Canada
Davide Tosi, Università dell'Insubria - Como, Italy

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

From Hardware Trace to System Knowledge – Data-intensive Hardware Trace Analysis

Andreas Gajda, Rainer G. Spallek
Department of Computer Science
Technische Universität Dresden (TUD)
Dresden, Germany
{Andreas.Gajda|Rainer.Spallek}@tu-dresden.de

***Abstract*—The capture of large amounts of hardware trace data by recent hardware trace units in System-on-a-Chip (SoC) defines the new requirements for hardware trace analysis. Several hundreds of MiB need to be processed in near real-time. The requirements on analyses are further increased with long-term trace capture and increasing frequencies of SoC. Hence, hardware trace analysis has become a data-intensive computing task. The article proposes an approach based on data, functional and pipeline parallelism in combination with data stream processing. As this is a 'work in progress', we will only outline the approach and the preconditions and decisions leading to the approach.**

***Keywords*-data-intensive computing; large data streams; data-processing pipelines; hardware trace analysis; analysis framework**

I. Introduction

Embedded Systems, also called System-on-a-Chip (SoC), facilitate multiple system units on a single chip. Current SoC consist of multiple processors, I/O units, graphic accelerators and highly specialized data processing units. Because most embedded systems cannot be easily debugged in-situ, vendors use special hardware units to gather information, so-called traces, of the system's behavior.

These hardware trace units are configurable devices, that record parts of the hardware context data and publish the records as a packet stream via dedicated hardware interfaces. Capturing the trace data influences the runtime behavior of the processor to varying degrees, depending on the kind of hardware unit. If the runtime behavior is not influenced by the trace capture, the trace capture is called *non-intrusive*. The captured context data contain the control and data flow, event counter data, and ownership identifier.

Trace data provide valuable insight into the dynamic behavior of the system, because non-intrusive data collecting in hardware concurrently with the execution represents the original system's behavior, which might have strict constraints on runtime. And it yields more detailed information about system events: the location, time, and order of their occurrence.

However, hardware trace units produce a great amount of data. These data must be transferred outside the chip by low bandwidth interfaces. This problem is addressed by the techniques of on-chip filtering, compression, and widening the bandwidth of trace interfaces. Nevertheless, observing the system's behavior over long periods of time results in a large amount of trace data. With future prospects of approximately 10^9 bytes of trace data to analyze, the recent trace data analysis, designed for small time periods, needs new approaches to yield results shortly after the data is captured.

The basic idea of our approach, is to preserve the natural format of a *trace stream* and apply all analyses as a *stream transformation*. The analyses aim to be highly parallel, because they are decomposed and arranged by data, function, *and* pipeline parallelism.

Of course, this requires the trace to express software level concepts that cannot be solved in current hardware trace formats. Software trace formats can carry this information, but their ability to carry hardware information is limited.

We will describe some distinctions and similarities of both trace formats (Section II), to motivate their combination into one trace format. The requirements of trace analysis (Section III) and the properties of a trace analysis framework (Section IV) will be described in order to provide a rough view of the idea.

II. Characterization of Trace

The properties of hardware (HW) and software (SW) trace are summarized in the format, content, and amount of the trace. We refer to the IEEE-ISTO 5001™ [1] standard (NEXUS) and Open Trace Format [2] (OTF) for examples of hard- and software trace. The same properties can be found in other formats as well (e.g., [3], [4]).

A. Trace Format

Hardware traces [3] are transmitted out of chip via data and signal lines. The data are submitted sequentially, such that they can be packed into a stream format, consisting of packets of variable length. The NEXUS standard already defines packets as ther basis of its message system. Software trace is written to file by instrumented software and can have several distinct formats, e.g., unstructured files, structured log files, or packet based software trace formats[1]. For comparison, we will only focus on the latter.

In packet based trace formats, the packet consists of a fixed size header and a variable packet payload. For some packets, the payload size is zero, because the header already

[1] In the literature, hardware trace packets are often referred to as 'messages', whereas software trace packets are referred to as 'records'.

 ISBN: 978-1-61208-188-5

TABLE I: Comparison of not-expanded raw (NEXUS) trace data and the same data expanded without (UTF) and with inherent compression (UTF_c) for the AutoBench benchmarks of the EEMBC [5] benchmark suite.

Benchmark	Covered Ticks (Million)	Size (MiB)			Emit Volume (Byte per Tick)			Expansion Factor	
		NEXUS	UTF	UTF_c	NEXUS	UTF	UTF_c	UTF	UTF_c
a2time	9.18	27.44	111.95	63.37	3.14	12.79	7.24	4.08	2.31
aifftr	28.86	88.47	440.99	178.50	3.21	16.02	6.49	4.98	2.02
aifirf	9.32	27.84	114.09	64.27	3.13	12.84	7.23	4.10	2.31
aiifft	27.93	85.53	424.58	172.10	3.21	15.94	6.46	4.96	2.01
basefp	9.19	27.56	114.46	63.34	3.14	13.05	7.22	4.15	2.30
bitmnp	9.20	27.20	110.94	63.38	3.10	12.64	7.22	4.08	2.33
cacheb	9.18	27.59	112.46	63.11	3.15	12.84	7.21	4.08	2.29
canrdr	9.22	26.45	112.90	64.14	3.01	12.83	7.29	4.27	2.43
empty	9.21	27.52	112.19	63.59	3.13	12.77	7.24	4.08	2.31
idctrn	9.22	28.04	119.62	62.62	3.19	13.60	7.12	4.27	2.23
iirflt	9.27	27.73	113.37	63.97	3.14	12.82	7.23	4.09	2.31
matrix	156.91	596.35	2447.58	1013.95	3.99	16.36	6.78	4.10	1.70
pntrch	9.20	26.96	110.10	64.00	3.07	12.55	7.30	4.08	2.37
puwmod	9.19	25.91	113.97	64.70	2.96	13.00	7.38	4.40	2.50
rspeed	9.22	27.21	112.89	63.96	3.09	12.84	7.28	4.15	2.35
tblook	9.17	27.36	112.41	63.49	3.13	12.86	7.26	4.11	2.32
ttsprk	9.23	26.93	113.43	64.25	3.06	12.88	7.30	4.21	2.39
average	20.16	67.77	288.11	132.75	3.36	14.29	6.59	4.25	1.96
median	9.22	27.52	113.37	63.97	2.98	12.30	6.94	4.12	2.32

carries all of the information. Trace formats typically utilize several kinds of compression, summarization, and conditional information. Conditional information is indicated by length fields or signaled by bits in the packet header and/or packet payload.

B. Trace Content

Most packets can be categorized into 'control flow', 'data flow', 'event counter', and 'run/trace control'. Hardware trace may contain additional packets for the manipulation of register and memory content. The NEXUS distinction of 'ownership trace' and 'device ID' is included in the categories of 'control flow' and 'data flow'. Software trace may provide additional packets for concepts above the software level, like peer to peer and collective communication.

Hardware trace content consists of memory and instruction addresses, device and thread identifiers, timestamp and tick counter information, memory content, break- and watchpoint events, and synchronization events.

Software trace content consists of numerical or text identifiers of processes, functions and variables, accessed variable value and performance counter value. Usually the content is ordered with respect to the execution, but timestamps or tick counter value provide an inherent (weak) ordering of concurrent processes and additional timing information for functions. Some trace formats allow global or interval based summaries.

C. Trace Amount

All formats support some kind of in-stream compression to reduce the bandwidth needs for transferred data.

In hardware trace streams, the used methods comprise transmitting differences to reference value, leading zero suppression, sampling and tagging of repeated data. Some hardware trace units use an on-chip filter, to limit the location and time of observation to points of interest. Software trace uses data reduction analysis, e.g., smart compression algorithms and sampling.

The size of a single packet in both formats is small and limited, but recording billions of events sums up to a few hundred MiB. See Table I for a few examples of NEXUS trace file sizes. Typically, hardware trace records a fine grained execution trace of the processors in a few seconds, and software trace records a coarse-grained program execution in minutes or hours.

The amount of the trace stream grows with the observation time and depth of analysis. One important use of hardware trace units will be the constant monitoring of the SoC, so-called long-term or endless trace, especially for near real-time analysis. Once available, the amount of data to process requires more computing power to extract the knowledge hidden in the data.

Experiences from working with the EEMBC [5] benchmark (Table I) show the median trace file size reaching 28MiB raw trace data for roughly 100ms of a single benchmark run. However, trace data volumes up to 600MiB were also retrieved. These trace data are compressed by any measures available in the trace format, including a branch based representation (i.e., *not-expanded trace*) of the control flow. When expanding the trace data to a single instruction based control flow trace (i.e., *expanded trace*), the expansion factor is about 4, which means dealing with about 115MiB of trace data.

Based on these data, transferring and analyzing the raw trace stream in near real-time, requires approximately 315MB/s bandwidth and data processing power for a single 100 MHz embedded target processor.

III. Hardware Trace Analysis

Trace stream analyses consist of filtering and analyzing the content. Filtering is the reduction of the trace events to a small

 ISBN: 978-1-61208-188-5

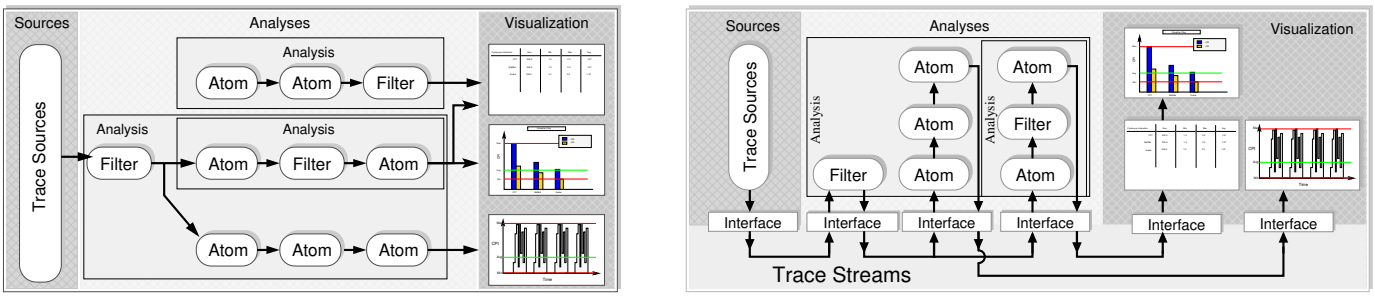


Fig. 1: Vertical and horizontal view of the analysis framework. The views are simplified, by not showing the interfaces between the atoms within an analysis. The arrows point in the direction of the data flow.

window of interest. This process is done at a minimum of two times: On-chip filter restrict the trace capture to code ranges, data ranges, or other user specified trigger events. These filter are set-up *pre-trace* versus *post-trace* filter, which reduce the amount of captured trace by suppressing trace packets. Post-Trace filter analyze the trace with similar criterion like pre-trace filter, but mostly more specific to the follow up analysis. Some filters are analysis inherent, like restricting the analysis to a certain kind of packets, e.g., those for data flow trace.

The analysis of hardware trace content maps the raw trace data back to different levels of the executing environment abstractions. For instance, analyses reconstruct the control or data flow at the instruction level, analyze the call graph at the function level and observe the scheduling behavior at the system level. Analyses may be built up on other analyses too, like measuring function execution times and calculating the statistical values afterwards. The structure of intermediate results might change, like reconstructing the instruction control flow from raw trace and mapping it onto control flow graphs. Intermediate results might be used multiple times, like creating the static and dynamic call graph from the software level function trace.

All (intermediate) results need appropriate structures and storage management, which provide fast access and large storage. Moreover, the intermediate and final results of the analysis need to be stored externally, e.g., for creating checkpoints or comparison with the results of other traces.

IV. Trace Analysis Framework

This section introduces the trace analysis framework. We will describe the framework and the decisions that lead to the structure of it, on an abstract level. We will introduce the aspect of analysis parallelization, the internal structure of the framework, and the internal trace format and provide the current state of implementation.

We call it 'trace analysis framework' to emphasize the aim of hardware and software trace analysis, using the same generic structure.

A. Parallelization

As shown before, the challenges of hardware trace analysis result mostly from the amount of data to process. Furthermore, since analysis is an interactive process, the period between trace capture and visualization of the analysis results must be short, since the developer might need to refine or restart the trace capturing process. Displaying intermediate results gives additional feedback of the analysis progress. In addition, the results from long-term trace are expected to be available *before* the trace has finished. Therefore, the trace analysis should be near real-time at least.

To achieve a speedup in data processing, the analysis task needs to be parallelized. The fundamentals of parallelization are the decomposition of data and function and the concurrent use of processing units to execute (different) functions on (different) data. Since all possible tasks for hardware trace analysis need to be decomposed for that purpose, this task might provide enough work for years.

Another approach is to create a parallelized analysis framework, which provides the means for complex analysis and the inherent decomposition of the analysis itself.

B. Framework Structure

The analysis framework (see Figure 1) shall exploit functional and data parallelism as much as possible. The analysis process is broken down into the functional units of the trace data source, analyses, and visualization. All the data exchanged between these components are transferred as trace data streams. All the data are encapsulated in one trace format, which is designed to host the hardware trace data and the analyses results at once.

The analyses are decomposed into atomic functions, which are also connected via trace data streams. Each atomic analysis has a minimal functionality. The atoms are combined into a complex analysis by connecting them via trace streams, where the output of an atom serves as the input to several other atoms. The analysis atoms might filter, transform, or enrich the trace stream, such that the output of the analyses can be visualized.

Since the atoms depend only on each other via their interconnections, the whole framework is highly parallel. The ability to duplicate the trace stream and use different atoms at once, using the same input data, provides a functional decomposition of complex analysis. The ability to duplicate and enrich the trace stream, provides additional data decomposition. Both kinds of decompositions are arranged via the

 ISBN: 978-1-61208-188-5

network, which is generated by the trace stream connections between the atoms.

The acceleration of the analysis is expected to come from the concurrent execution of different analyses at once. The input trace stream and any intermediate trace data can by duplicated and serve as input to several subsequent atomic analyses, which extract the different properties of the data concurrently. The independence of the atoms allows concurrent execution, only with synchronization at the input and output channels.

The visualization of the intermediate results benefits from the concurrent handling of raw, intermediate, and final trace data. If the visualization receives the first result, the first atom still processes trace data from the trace source.

The framework is able to balance the workload by examining the amount of data transferred between the atoms. As long as there is no input for an atomic analysis, the task does not need to be processed. Since this enables the dynamic switching between active and dormant state of atoms, this might lead to further optimizations.

We understand that a complex analysis might not be decomposable, or at least not without introducing significant overhead. In this case, the analysis can be treated as an atom itself, without the loss of generality of the framework.

C. An Analysis-oriented Trace Format

Several reasons lead to the design of a trace format for the framework's internal use: At first, there are several hardware trace formats defined and used by different vendors. Supporting all of these formats for all analyses is an expensive task. Second, hardware trace formats are *bandwidth-oriented*. They invoke several compression techniques, reducing redundant or otherwise reconstructible data from the transferred data, to decrease the bandwidth requirements for on-chip trace hardware. While this is very desirable for communication purposes, it is a large overhead creating task for trace analysis, to expand the trace into an analysis-usable format. Third, hardware trace represents events, control, and data flow at the hardware level only, which is fine for the purpose for which they were defined, but makes it impossible to represent software level trace. Software trace formats were designed to represent, e.g., the control flow at the function level and, therefore, lack (efficient) support of control flow trace at the instruction level.

These are enough reasons to create a new *analysis-oriented* trace format, which

1) is structurally compatible with the existing hardware traces,
2) provides a framework to host analysis results,
3) provides a framework to host a software level representation of hardware level events, and
4) has a low overhead impact on analysis tasks.

We call it *'Universal Trace Format' (UTF)*. It is packet based, provides a means for the representation of control and data flow at the hard- and software level, multidimensional event counter and control packets for trace formatting, messaging and other events. The definition space for new packets is still extendible, such that new concepts can be introduced in later versions.

Since we expect trace stream transfers to the hard disk and over networks, the trace format includes an inherent lossless compression, which reduces the transferred number of bytes. To avoid a large computational overhead, the compression algorithm is kept simple. Table I shows the reduction of the trace file sizes, when using the compressed trace format.

D. Work in Progress

The framework exists in an experimental stage, with a few atoms and analyses implemented. In this stage of implementation, we focus on the proof of concept. While we are already testing complex trace analysis, no results are published yet. The first implementations focus on instruction level analysis, including instruction covering, control flow analysis, and timing analysis. The next steps include block level analysis such as call stack and call graph visualization.

Looking beyond this framework, the visualization of the results needs to be included. To date, there are only textual representations of the trace data, but we know the importance of the graphical representation. We imagine a generic framework, which will be fed from the trace data and provides generic visualizations of flow graphs, counter time lines, diagrams, and animations.

V. Conclusion and Future Work

This paper has provided a view of our parallel trace analysis framework. We presented the preconditions of the analysis and the resulting conclusions. Hardware and software trace properties were explained and the implications for the combination into a single trace format given. The internals of the framework were outlined and an outlook of future development directions were given.

The framework still has to prove its usefulness, but we expect performance results soon. Further research in parallel trace analysis and performance improvement is possible and encouraged. We also imagine extending the framework for analyses at abstract levels, e.g., the 'runtime verification' in the field of model checking.

VI. Acknowledgments

The authors gratefully acknowledge the financial support of the European Union and the Free State of Saxony.

References

[1] (2011, Oct) Nexus 5001 Forum™ Standard. Website. [Online]. Available: http://www.nexus5001.org/standard

[2] (2011, Oct) Open Trace Format – Homepage. Website. [Online]. Available: http://www.tu-dresden.de/zih/otf

[3] C. MacNamee and D. Heffernan, "Emerging on-ship debugging techniques for real-time embedded systems," *Computing Control Engineering Journal*, vol. 11, no. 6, pp. 295 –303, dec 2000.

[4] (2011, Oct) Introduction into ParaVer. Website. [Online]. Available: http://www.bsc.es/ssl/apps/performanceTools/files/docs/intro2paraver_MPI.tar.gz

[5] (2011, Oct) The embedded microprocessor benchmark consortium. Website. [Online]. Available: http://www.eembc.org

 ISBN: 978-1-61208-188-5

Grid Spider: A Framework for Data Intensive Research with Data Process Memoization Cache

Daichi Yamada, Tomohiro Sonobe, Hiroshi Tezuka and Mary Inaba
Graduate School of Information Science and Technology
The University of Tokyo
Tokyo, Japan
{*yamada.daichi, sonobe.tomohiro, tezuka.hiroshi, mary*}*@ci.i.u-tokyo.ac.jp*

***Abstract*—As computational power grow, the new field of "Data Intensive Computation" has emerged in which vast amounts of data generated by radio telescopes, particle accelerators, electron microscopes, genomics and Earth observation equipment is processed. In most cases, once the data has been accumulated, it is not overwritten. It has also been observed that in many cases the very same software is used to pre-process the very same data, leading to identical results. To address these issues, we propose "Grid Spider", a framework for data intensive scientific research which is optimized to avoid re-computation through the utilization of our file cache mechanism called "Data Process Memoization Cache" or DPMCache. This mechanism requires pre-processing applications to maintain referential transparency. Both the data and the application are registered with Grid Spider prior to processing, and for each execution of the application, Grid Spider records the history of the coupling of the application, the input data fie, and the output data file. To evaluate Grid Spider, we have implemented "GEO Grid Spider II", which is a framework within which geo-scientists can evaluate satellite data archives.**

***Keywords*-data intensive; memoization; file cache; cache replacement;**

I. INTRODUCTION

Computers have been playing an important role in the progress of scientific research from their inception. As computational power has risen, supercomputers have been increasingly utilized to provided the number-crunching necessary for scientific simulations. More recently with the rapid development of storage devices and distributed computing, "Data Intensive Computation" has emerged which analyzes huge volumes of of scientific observation data.

Radio telescopes, particle accelerators, electron microscopes, genomics and Earth observation equipment are typical sources for such data requiring intensive computation [1]. In most cases, the observation equipment tends to be quite expensive, and many researchers and research groups often share the data.

One common characteristic of this sort of data is that it is never overwritten. Once written, it is merely read by the interested researchers. In addition, it has been observed that, in many cases, the very same software such as code libraries for pre-processing is applied. Assuming that the very same software is processing the very same data, is reasonable to conclude that the results will also be the same. If a method of avoiding this kind of re-computation were possible, it would improve the resolution of the results.

To this end, we propose "Grid Spider", a framework tailored to data intensive scientific research. Grid Spider requires applications to maintain their referential transparency through a file cache mechanism called "Data Process Memoization Cache" or DPMCache, thereby avoiding to a high degree re-computation. Grid Spider users are scientific researchers who heavily rely on application programs, but seldom write their own code. Prior to processing, both the data and the application are registered with Grid Spider. Grid Spider then controls the execution, and records the coupling of the application, the input data file, and the output data file. To evaluate Grid Spider, we implemented "GEO Grid Spider II", a spacial data mining framework primarily utilized by geo-scientists evaluating satellite data archives.

The organization of this paper is as follows. In Section II, we propose a file cache mechanism called DPMCache, which provides the core of the Grid Spider framework. We then propose the Grid Spider framework, our core concept. In Section III, we discuss GEO Grid Spider II, which is a framework for the processing of satellite data by geo-scientists. In this section we will introduce some interesting results. In Section IV, we provide an evaluation of the Grid Spider framework through simulations, offer related work in Section V, then draw relevant conclusions in Section VI.

II. DATA PROCESS MEMOIZATION CACHE AND GRIDSPIDER

In 1968, very early in the era of computers, memoization [2] was proposed by Donald Michie to optimize computation by reducing the cost of re-calculation; for each function call, an argument and its result is recorded, and the result is used without the computation in subsequent calls to the same function with the same argument. This technique is used in many fields, and the condition for a function for which memoization works properly is, the function always returns the same value for identical arguments. This technique was primarily used in the functional language, but later in [3],

 ISBN: 978-1-61208-188-5

Peter Norvig shows that, if only the function has referential transparency, then external automatic memoization is possible. This technique has been applied in many fields [3] [4]. We noticed that this concept can be also applied to the application programs for data intensive science in which applications and vast data archives are shared by many researchers, and that, once the data is acquired, it is never overwritten. We extend the concept of memoization beyond the domain of a single program execution to multiple program executions using a file cache. For this, we need to allow programs to have referential transparency to store the output of the applications and to record the triplet of (1) the application used, (2) the input file applied, and (3) the output file generated. Since the cost of storage is continuously dropping, it is now feasible to divide an application into several modules and store the intermediate results of each module. This will enhance the share-ability of the application.

We propose a file cache system we have called "Data Process Memoization Cache" or DPMCache. An application consists of a sequence of application modules which have referential transparency, and the output of each module is stored as a cache file until storage space is full (Figure 1). When storage becomes full, cache replacement occurs with a replacement policy that is dynamically changeable for the DPMCache based on the history that its framework records.

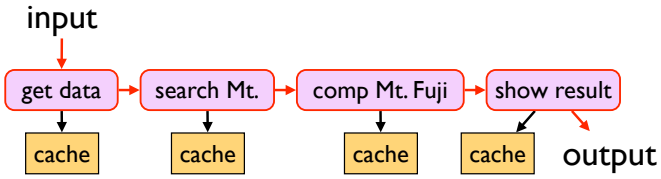


Figure 1. Application list

It is often observed that several applications use identical pre-processing, and in these cases, these applications constitute the application tree (Figure 2). The application tree is a union of the application lists, and the results of shared modules are also shared. For example, processes such as distortion correction and noise removal are required my multiple applications.

We propose the Application Tree Structure Cache Replacement Policy (ATS). The ATS scores each cache file based on the Application Tree Structure. Modules with many branches in an application tree are likely candidates for shared pre-processing. To detect this, we adopt a scoring system in which (1) cache files generated by a module with many branches, (2) cache files created by a module requiring more time, and (3) cache files of a smaller size are all given a higher score. Therefore, cache files with lower scores are selected to be replaced by ATS. Figure 2 shows how ATS

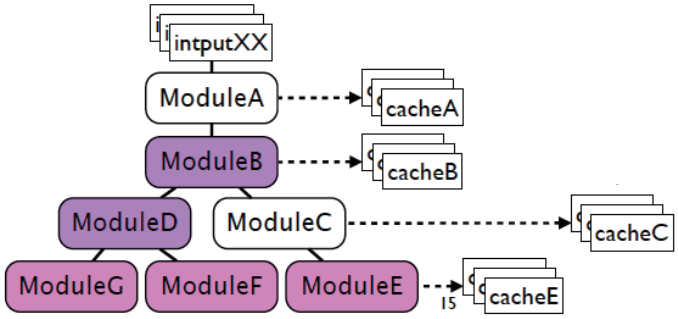


Figure 2. Application Tree

leaves untouched files created by Module B and Module D which score higher due to their many branches.

To utilize this DPMCache scheme, we propose a framework for data intensive scientific research called "Grid Spider" which has the objective of omitting avoidable recomputation by sharing the results of all users of the framework. Grid Spider controls (1) the enforcement of the referential transparency of application modules, (2) the management of data files so that files are unique and are not overwritten, (3) recording the history of history triplets consisting of the module, the input file and the output file, and (4) applying the optimal cache replacement policy to cache files based on the history triplets.

Application modules and input data are registered with Grid Spider in advance, and users construct an application by combining application modules.

Grid Spider is intended to be used in a distributed environment. Figure 3 shows the configuration of the system. The users send an query to the parent node consisting of search areas and applications. The query is then divided into requests that correspond to the divided search areas, and these requests are assigned to child nodes. The child nodes run the application, and they return the results to the parent node. The application tree and the cache files independently correspond to the the child nodes. The assignment of the request depends on the coordinates of the search area, and each child node is assigned its own search areas. If there is a conflict between the assignment nodes, the parent node takes over the load-balancing.

The optimal replacement policy is determined by how the cache is accessed. For example, LRU works well in many cases, but it works poorly when scanning arrays. Therefore, we propose Node Scoring Adaptive Policy or NSAPolicy which dynamically switches the replacement policy in a distributive environment. In our proposal, child nodes send the cache hit rate to the parent node, and the parent node chooses the highest scored policy and applies it to the child nodes. The parent node checks the highest score policy and switches the child nodes's cache replacement policy.

 ISBN: 978-1-61208-188-5

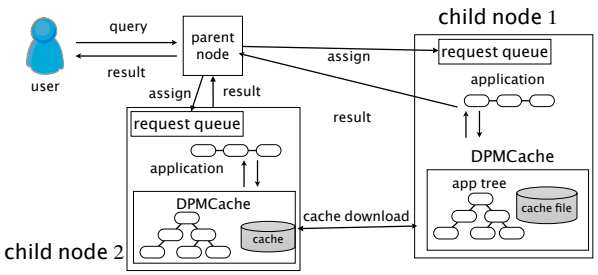


Figure 3. System design

III. GEO GRID SPIDER II

Using Grid Spider, we designed and implemented GEO Grid Spider II (GGSII) as an instance of the Grid Spider framework for geoscientists treating satellite data.

On GGSII we can easily implement search applications using geographical information. Figure 4 shows the example results of a search for circular objects on global aerial maps such as round irrigated farm plots. Figure 5 shows a sample result from a search of face-like objects. This application was implemented by modifying the sample code in OpenCV. Thus, we can easily incorporate other applications into the framework as modules.

Figure 4. Irrigated farm plots in Brazil

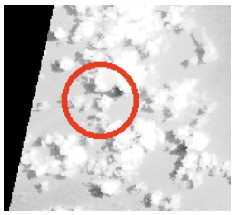


Figure 5. Face search application (Venezuelan coast)

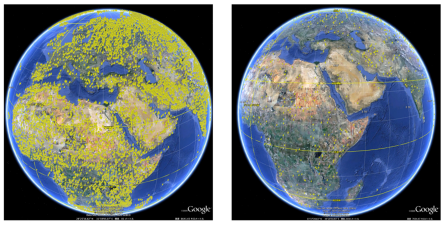


Figure 6. Detected mountains with higher threshold (left) and lower threshold (right)

Figure 6 shows the locations of the similar mountains detected by the algorithm [5]. The left photo depicts the results of the first trial, and the right photo depicts results after the threshold adjustment. In cases where threshold and parameter tuning is desired, the GS framework works quite well.

To make this framework useful to active geoscientists, we consulted them and considered several use cases. In the consideration of distributed system implementation we categorized these use cases into three groups: (1) **world wide** : involving global searches for objects with specified features; (2) **concentrated** : involving focused searches after disasters such as large earthquakes or volcano eruptions; (3) **mixed** pattern : combining both global and focused elements.

IV. EVALUATION

For the evaluation of GGSII, we first performed the simulation, then implemented the system for clusters.

We compared the replacement policies (1) First In First Out or FIFO, (2) Least Recently Used or LRU, (3) Segment LRU, (4) Bimodal Insertion Policy or BIP, (5) the proposal method ATS, and (6) the proposal method NSAPolicy.

A. Simulation

We implemented a event-driven type simulator in Java where the parent-node/child-node ratio is 1:20.

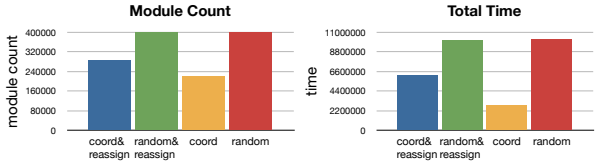


Figure 7. World Wide, Global survey

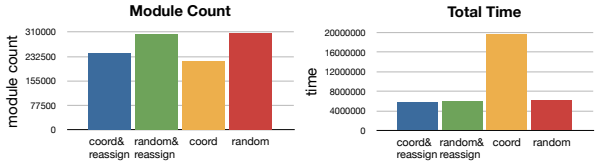


Figure 8. Concentrated, disaster

For the distributed system, we compared the combination of location specific distribution and random distribution against the combination of with reassignment and without reassignment. Figures 7 and 8 show the module counts and the total time for the global distribution use case and the local disaster distribution use case.

Random distribution proved unproductive due to the request being assigned to a node lacking a cache file. Location-specific distribution was more productive in respect to cache hits, but experienced distribution imbalance, especially in disaster cases. We adopted the combination of first location specific distribution migration for load-balancing.

Figure 9 shows the changes by total cache size. This graph show the use case of a world wide survey. ATS performs well.

 ISBN: 978-1-61208-188-5

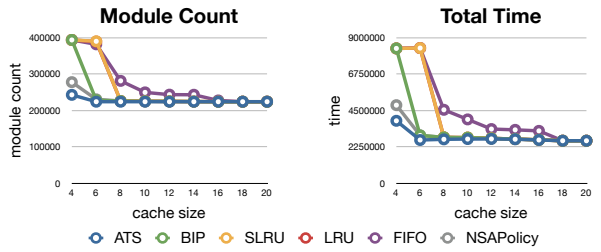


Figure 9. Changes by the total cache size for the intensive global case

B. Experiment on cluster system

We implemented the GGSII on a cluster of eight PowerEdge R410s with an Intel(R) Xeon(R) CPU E553 with 2.40GHz of 4 core x 2, with a 12GB memory and a 430GB HDD. For data, we used Shuttle Radar Topography Mission data or SRTM, and Digital Elevation Model or DEM. Each file was 2.8MB, and the 14,168 files corresponding to almost 80% of the Earth's surface totaled about 40GB. We repeatedly ran the application to detect mountains similar to Mt. Fuji as described in section III, and changed the parameters for each execution.

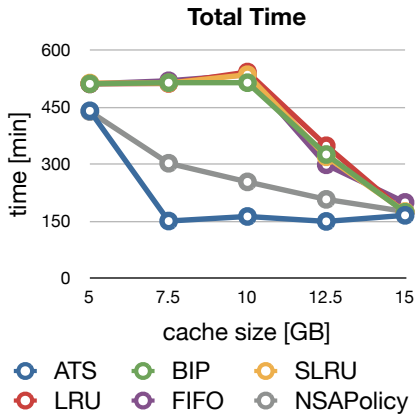


Figure 10. Comparison of query processing time (survey pattern)

Figure 10 shows the results from an experiment similar to the one depicted in Figure 9. ATS performs well to a level comparable to the simulation results.

V. Related Work

A. GEO Grid

Global Earth Observation Grid or GEO Grid [6] is a grid system that was developed by the National Institute of Advanced Industrial Science and Technology (AIST). The application provides through GEO Grid a birds-eye view displayed in the viewer in 3D and in the Volcanic Gravity Flow Simulation. For example, the GEO Grid task force released the crustal movement data and the propagatin animation of the tremors [9] recorded during the 2011 Tohoku Earthquake on Japan's Pacific coast.

B. Cache Consistency

Maintaining cache consistency is important to a cache system in a distributed environment. "If-modified-since" [8] is used on Web systems, and is used to disable one cache when the record is updated on a second cache on a distributed file system such as CODA[7].

In addition, these systems often use read-only data so consistency is maintained. For Gird Spider, the input is write-once, and the cache is write-once because the module has referential transparency. Therefore, cache coherency is automatically maintained.

VI. Concluding Remarks

We are proposing the file cache scheme DPMCache and a framework for data intensive research called Grid Spider. To demonstrate the usefulness of Grid Spider, we implemented GEO Grid Spider II which focuses on satellite data archives and geophysicists. We evaluated the system using both simulation and real implementation, and compared the replacement policies. We have shown the effectiveness of our replacement policy which utilizes the history of the execution of the applications.

For future research, we intend to focus on the time in queue of disk IO, and the communication between each node. In addition, we plan to utilize the scheduling algorithms to reduce processing time.

References

[1] Microsoft Research, "The Fourth Paradigm: Data-Intensive Scientific Discovery", in T. Hey, S. Tansley, K. Tolle, 2009.

[2] D. Michie, "Memo Functions and Machine Learning", Nature, No. 218, pp. 19-22, 1968.

[3] P. Norvig, "Techniques for Automatic Memoization with Applications to Context-Free Parsing", Comput. Linguistics, Vol. 17 No. 1, pp. 91-98, 1991.

[4] J. Mayfield, et al., "Using Automatic Memoization as a Software Engineering Tool in Real-World AI Systems", Proc. of the 11th Conf. on AIA, 1995.

[5] R. Wakuta and T. Sonobe, "SPGF Search Places by Geographical Features all around the world Search and verification system", SAINT2010, 2010.

[6] S. Sekiguchi et al. "Design principles and IT overviews of the GEO Grid", Syst. J., IEEE, 2008.

[7] J. Kistler and M. Satyanarayanan, "Disconnected Operation in the Coda File System." ACM Trans. Comp. Syst., 1992

[8] R. Fielding et al., "Hypertext Transfer Protocol – HTTP/1.1", RFC2616, June 1999

[9] GEO Grid disaster task force, http://disaster-e.geogrid.org/, Jan 2012

 ISBN: 978-1-61208-188-5

Native-based RIA Development Using Mobile Phone Methodologies to Improve the Overall Project Functionalities

Jonathan Bar-Magen Numhauser
Dept. of Computer Science
University of Alcala
Alcala de Henares, Spain
jonathan.bar@edu.uah.es

Jose Antonio Gutierrez de Mesa
Dept. of Computer Science
University of Alcala
Alcala de Henares, Spain
jantonio.gutierrez@uah.es

Abstract—**In this article, a work at progress will be exposed on a study that is being elaborated by our investigation team regarding the working methods of Resource Intensive Applications (RIA) development. Using previous experience in software development for limited devices and applying our previous investigation work on collaborative component based development for web frameworks this work elaborates a complete workflow for RIA development, resulting in an improvement of the RIA functionalities and work development time. These improvements derived in an enhancement of the developing process by reducing time and increasing component recycling.**

Keywords*—*Methodology; Optimization; Workflow; Collaborative; Benchmark.

I. INTRODUCTION

The following article will expose in details a development methodology that originally was oriented to mobile phone's software development. The purpose of this is to obtain circumstantial statistic information on the functionality of Resource Intensive Applications (RIA) application that apply these methodologies, and as a consequence analyze the possible improvements to the overall development process that may allow the creation of an alternative working methodology.

Mobile phone's software development is well known in the computer science field. The most significant technology to cover this working environment was the Java 2 Micro Edition (J2ME) [11]. For many years, such a technology was used for mobile phone's software development, as well as other smaller technological environments like Symbian and Research in Motion (RIM) [12].

Apart from the available programming languages and the virtual machines that allowed the programs to execute correctly, to have a successful impact on the mobile phone's market, there was an urging need to cover the largest amount of targeted devices, and depending on the device's characteristics, the number of possible versions of the same program could have resulted on an average of 30 different versions.

The creation of a methodology to reduce the developing cost was in place and as consequence such methodology was designed, coming hand in hand with tools that allowed the correct functionality of the established steps [1]. The results were significant reducing the cost in time and work effort of software development for mobile devices.

This success was rapidly studied and as a result derived in the study of its availability for other technological fields, in particular RIA development, which is being explained in this document. The work in process described in this project has taken place for the last year, studying many aspects of the software development life cycle, applying it on real cases, and collecting the results for further analysis. It aims to expose this investigation work, and its future objectives.

The paper will initially describe the environmental pre-requisites, following a detailed explanation of the available components as well as the problem at hand, and ending with the exposing a few results of the practical experiment.

II. LIMITED RESOURCES DEVICES METHODOLOGY

As it was stated in the introduction, the motivation for this investigation work is to reduce considerably the development time of RIAs as well as improve their functionality to make the resulting software optimal for network based activity.

At first it was decided, based on experience and statistic results to promote the native software development, which consists in adapting the resulting software application to each hardware, in contrary with developing in automatic multi-platform adaptability, originated from the Java basic motto: "Develop it once and run it anywhere" [10]. This significant decision in the approach may not seem optimal, but we can confirm that this new limitation opened other alternatives and new needs to ensure the software functionality and quality in the targeted devices.

RIA development in the present day is as diverse as the mobile phone's market. To ensure maximum impact on the market RIA projects require the developers to manufacture a PC version as well as a mobile version, which include mainly 3 different dominant technologies, Android, Cocoa and RIM [12].

 ISBN: 978-1-61208-188-5

Regardless the specific native technology, the development in each different technology is what will ensure a correct and fully powerful functionality of the resulting RIA. The project will cover the largest amount of devices and will allow an optimized feedback.

Software development for limited resources devices forces the developer to pay more attention to details, improving his development skills and requiring the highest experience to ensure that the application will correctly work with the variety of limitations. One of the main obstacles in the current market is the large amount of flaws in the resulting applications, and even though the common excuse is the difficulty of developing in such devices, there is no excuse in not using the past experience to avoid repeating those same mistakes. The definition of a methodology intends to offer a solution to this problem.

Some of the characteristics that form this optimized methodology consist of debugging, which may sound trivial, but considering the limitations of each device it end being more complicated. Not to mention that the debugging requires passing through 2 experienced developers. Debugging and testing are strongly linked, and as more documentation is available, the better [2].

Documentation is another characteristic of this methodology. Once a device is fully documented, and a great database with all the limitations, hardware bugs, and tricks is established, the developing process will reduce its dependency on experienced developers and allow a high level development for more less experienced developers.

An example of this characteristics are either JMEPolish, which offers a consistent online mobile devices database with many of the possible flaws and problems a developer may encounter, as well as newer databases created by institutions like Adobe for graphical based development in devices, or Eclipse for raw JME development.

This same process will be adapted on a greater scale for the methodology of a RIA development, to improve the collective knowledge.

Another characteristic is the creation of common libraries, and the use of code pre-processing tools to allow a more dynamic composition of resulting applications, and so reducing the great effort of code modification in every existing version of the RIA.

As a result of applying this methodology, the impact was significant on mobile phone devices. The development process before the existence of such methodology, on a practical tested enterprise, was the creation of a base version on a specific device, followed by its adaptation on the variety of devices to which the software was targeted. The process was based on creating for each device an independent version, increasing the possibility for mistake and reducing the developer time to add new functionalities. The lack of resources affected the time the developers had to add and improve the common development library as well as improving the feedback in documentation.

Following the implementation of this methodology, and the consequent improvements based on experimental data, the process time dropped significantly, to a 300 percent average, in an exponential curve that stabilized. The program optimization to each targeted device was also significantly improved, allowing the programs to run fluently.

Figure 1 describes a resulting developing process based on the established methodology.

III. RIA NATIVE DEVELOPMENT METHODOLOGY

As a result experience from the mobile phone devices methodology and the gradual evolution of the RIA market in the last few years, we stumble upon the fact that much of the used characteristics in the mobile phone device methodology could be migrated to RIA development considering the new network types and its application on a broader devices network.

The market has evolved to include a various number of electronic devices that come along with its specific native technology, which will serve us greatly if we applied the already developed and tested methodology detailed in the previous section.

Our work in the actual phase consist in applying the methodology developed for mobile device on RIA development of a prototype program, so we can evaluate the time, functionality, and improvements to the methodology as well as the organization that lays behind it.

While in limited devices software development we had to consider as the highest priority the limited resources, in RIA we have to reorder the priorities to turn the development on multiple devices as a united modular project resulting in managing the RIA as a compendium of modules [4]. Those modules represent many of the functionalities required from the RIA, and each will have in its time a representation in a device.

The possible structure may be described as followed: (Figure 2)

- RIA structured as a collection of modules.
- Module is a collection of version modules.
- Each version module is a representation of the module on a specific device.

The creation of a united database from which the members of the development team may acquire information will be similar as in the case of limited devices development. The difference will consist that in the RIA case we seek to cover more than limited devices, but also a variety of devices from PCs to TVs and etc.

The impact on the functionality of each module version will be proportional to the experimental experience joined by the documental database gathered. Our speculations point that an improvement will be most significant on the optimization level of the program allowing the RIA to function substantially better and take advantage of the maximum resources accessible in each device for the RIA.

 ISBN: 978-1-61208-188-5

IV. INVESTIGATION WORK OBJECTIVE

One of the reasons for which we started with the study of this methodological alternative is because of our past experiences with RIA applications. From this we have noticed in many fields of work a lack of organization, and we couldn't find an already established agile methodology to apply in our case.

We have been working with SCRUM [3], and a developed collaborative components methodology that was design as well by our team, but still it was mainly oriented to light applications and not intensive.

Our team has also worked on mobile phones methodologies, and we saw a much more adequate methodology in these than in the agile alternative. The main reason is because in this case we are dealing with multiple devices for RIA, which require by itself a custom made organization.

So the objective for which we are working is to obtain metric information on the impact of this kind of methodology over RIA applications. To obtain a possible derivative of the mobile phones methodology as well as the collaborative components methodology, that will permit the establishment of a more adequate working method for the development of RIA.

The wished result is that the methodology will allow the investigator to begin a development work on a RIA, but underneath the RIA will be oriented and targeted to a broader network of devices, which will form a feedback network and improve the functionality of the RIA by taking advantage of the maximum capacity of each device in the network (native based development).

V. EXPECTED METHODOLOGICAL STRUCTURE

The resulting methodology is expected to form a dynamic structure that will allow a constant modification system. In addition to the standard working methods that abound, this methodology pretends to depend on a number of tools and standards that will establish the necessary steps to successfully create a complex RIA.

In the assumption that the basic steps of development, analysis, design, and maintenance will be fulfilled, we consider the need to add some basic roles, to ensure the quality of the methodology [5].

To implement a rich and optimized documentation database a documentation expert should be considered for the workflow. The role of Documentation Engineer is a possible option to certify the quality of the resulting support documentation and its administration.

Developers with experience in limited resources devices are recommended for the development steps of the working method. Their experience is always welcome, and their developing methods will significantly affect the resulting optimization of the RIA.

In addition, we are studying the possibility of adding collaborative based roles, to improve the recycling of modular components [6]. Being the case of RIA, and the necessity of ensuring the development quality of each module and its module versions, the sharing of existing modules and distribution of the developing work, reducing the developing time and allowing a more detailed feedback of the work taken place will improve the functionality results of the upcoming RIAs.

An international communication manager (ICM) as well as a global project manager will be an adequate incorporation to the working team [1].

Tools to control the versioning of the RIAs have been developed and are now on prototype stage, but can be confirmed as very useful and growing in importance inside the workflow.

VI. DOCUMENTATION DEPENDENCY

Documentation in our upcoming methodology will have a main role. Its impact on previous mythologies stated in this work, in specific the mobile devices methodology had seriously effect on the resulting data, improving the overall work process, and allowing the users of this work process to adapt its steps in a variety of targeted software.

The reason for adopting this step, regardless the extra resources needed to invest in it consists in the need to maintain a certain control on the manufacturing of the RIA product. Eventually even if in this particular case, the documentation process may take place depending on the targeted devices; it is always recommended, based on our previous experience on mobile methodologies, to execute an exhaustive documentation process, which will allow a rich feedback for both the methodology as well as the project life cycle.

VII. OVERALL PROGRESS

In the last year of work, the results were the following. The first step of achieving an initial prototype for modules and modules version was successfully completed, and in consequence we achieved favorable results of RIA modules operating on a variety of devices.

A structural design was established for the creation of such RIA projects, complying with the need for documentation and first steps of the resulting new work method. The technology and previous methodology required us to adapt the development steps to comply with the characteristics of a RIA project, and a first glance of this was successfully achieved not long ago.

Our effort to achieve a unified local development environment for native software development began to show a logical structure. Adding to this environment the development rules established for each module and native context, enforced a number of changes to the original hypothesis that had to be applied to comply with the overall compatibility.

Following the defined development structure in Model-View-Controller MVC, each environmental technological

 ISBN: 978-1-61208-188-5

context (layer) covered a certain percentage of the Model, the View and the Controller modules.

Database information would be treated under a database native technological script, example MySql. This technological context would be limited to Model treatment, even if database scripts in general can manage Controller actions, this action was associated with the management script on the server side, example PHP scripting technology.

Both database and script management technologies will be treated on the server sides, and will comply with the overall Model and Controller requirement of a global project in our methodology.

On the Client side we will end up having the intensive application, RIAs that will comply with the View and Controller requirements of the global project.

This general structure will ensure an independent technological development for each native context, and as a result allow the creation of a component based development methodology, requiring the expertise of members in each of the native context, and establishing a successful communication channel on a technological level between the layers as well as between the members of each layer.

The development requirements will be the core on which our methodology will thrive, complying with the general characteristics of an Agile Methodology, basing most of the productive work on the developing teams.

The use of limited resources development methodologies in each of the layers will result in a trust worthy system, improving the execution result in each layer individually, and enhancing the overall result of the global project in general.

These results were documented, and allowed the creation of a working connection between the lately known IOT "Internet of Things" development methodologies and RIA based development methodology, establishing that for each native system in its appropriate development context by combining those systems in an unified global project using the necessary roles and developing steps acquired from the previous investigation works, the reach of the overall projects increase substantially[5].

Possible example devised from such connection may be RESULTA, a project developed in the context of the ministry of industry in Spain with the purpose of improving the companies work effort in general developing project. RESULTA was a project developed in multiple entities, and on multiple platforms and native contexts, and successfully we were able to combine the multiple scenarios and create a unified project applying the concepts detailed above.

At present, we are proceeding with the enrichment of the modules and the modules database, the creation of a feedback system to allow upgrades and the establishment of other development steps in the out coming methodology.
We expect to achieve a working prototype that complies with the steps achieved in the following months, we had as an objective to achieve it during the month of November, but we encountered some structural difficulties, which will postpone our results.

ACKNOWLEDGMENT

The University of Alcala for facilitating its installations.
The OTRI of the University of Alcala for allowing the transition of knowledge for practical use.

REFERENCES

[1] Bar-Magen J., Gutierrez Martinez J., De Marcos L. and Gutierrez de Mesa J., 2010, Colaborative and Component Based Software Development Methodology, *Proceedings of the IADIS International Conference Applied Computing*, Rumania, pp. 287-290

[2] Griss M. L., 1997, Software Reuse Architecture, Process, and Organization for Business Success, *Proceedings of the Eighth Israeli Conference on Computer Systems and Software Engineering*, Israel, pp. 86-98.

[3] Murphy C., 2004, Adaptive Project Management Using Scrum, *Methods & Tools Global knowledge source for software development professionals,* Volume 12 - Number 4, pp. 10-23.

[4] Menkovski V., Exarchakos G., Liotta A. and Cuadra-Sánche A., 2011, A Quality of Experience management module, *International Journal on Advances in Intelligent Systems*, Volume 4 Number 1 & 2, pp. 13-19.

[5] Grambo G. and Oberhause R., 2011, Contextual Injection of Quality Measures into Software Engineering Processes, *International Journal on Advances in Software, vol 4 no 1 & 2*, pp. 76-99.

[6] Ribaud V., Saliou P., Saliou P. and Y. Laport C.,2011, Towards Experience Management for Very Small Entities, *International Journal on Advances in Software, vol 4 no 1 & 2*, pp. 218-230.

[7] Gargantini A., Riccobene E. and Scandurra P.,2010, Combining Formal Methods and MDE Techniques for Model-driven System Design and Analysis, *International Journal on Advances in Software, vol 3 no 1 & 2*, pp. 1-18.

[8] Fross M. Wake, 2003, How to Select a QA Collaboration Tool, *Methods & Tools Global knowledge source for software development professionals*, Volume 11 - Number 1, pp. 26-31

[9] Ponder M., 2004, Component-Based methodology and Development Framework for Virtual and Augmented Reality Systems Unified, *École Polytechnique Fédérale de Lausanne*, Rumbaugh

[10] Dabbs Halloway S., 2002, Component Development for the Java(TM) Platform, *Addison-Wesley Longman Publishing Co.*, Inc. Boston, MA, USA.

[11] Barchino R. Experiences in applying mobile technologies in an e-learning environment. *International Journal of Engineering Education.* 2007;23(3):454-459.

[12] Chia-Chi Teng. Mobile Application Development: Essential New Directions for IT. Information Technology: New Generations (ITNG), 2010 *Seventh International Conference on 2010*:471.

 ISBN: 978-1-61208-188-5

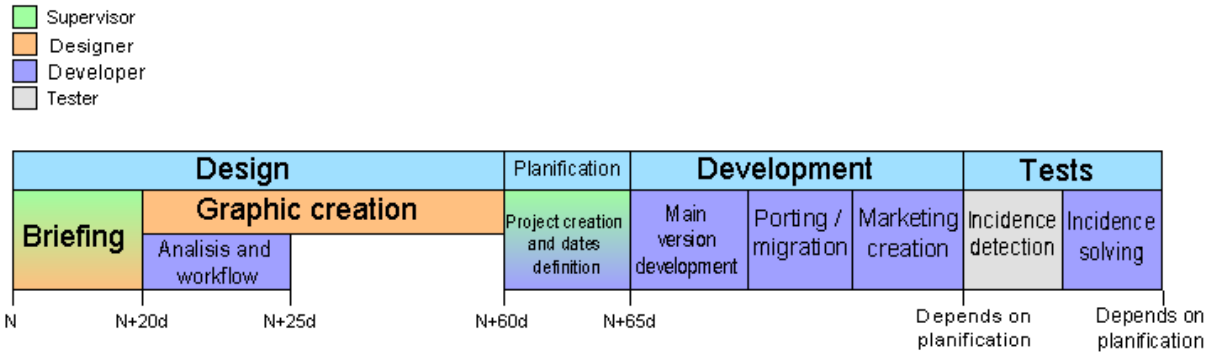


Figure 1: Practical workflow for a prototype RIA based on optimized methodology development.

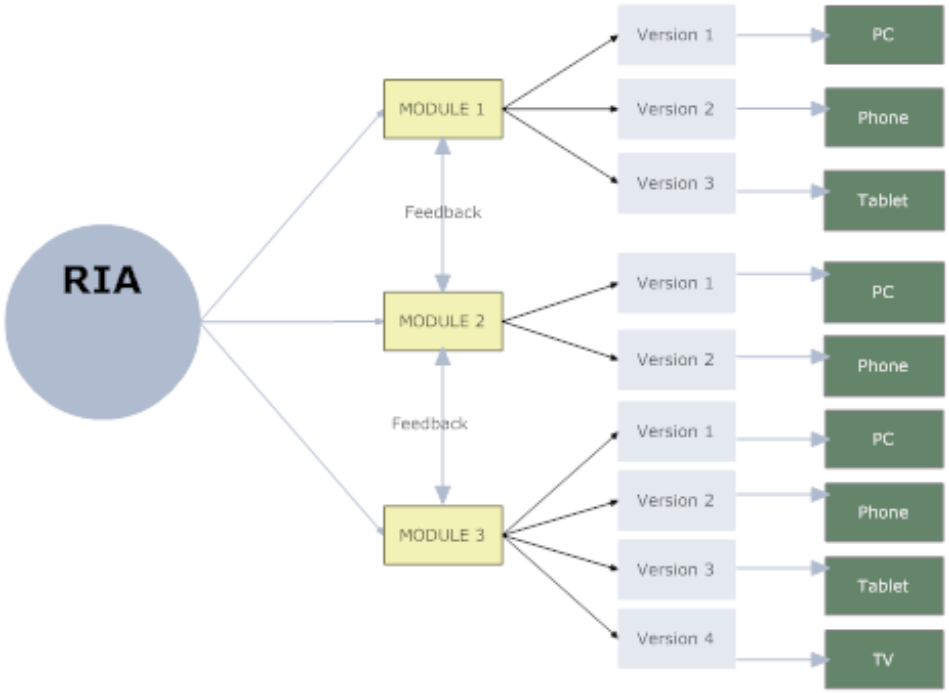


Figure 2: Modular distribution of an RIA.

 ISBN: 978-1-61208-188-5

Mining Interesting Contrast Sets

Mondelle Simeon Robert J. Hilderman Howard J. Hamilton
Department of Computer Science
University of Regina
Regina, SK Canada S4S 0A2
{*simeon2m, hilder, hamilton*}*@cs.uregina.ca*

***Abstract*—Contrast set mining has been developed as a data mining task which aims at discerning differences across groups. These groups can be patients, organizations, molecules, and even time-lines. A valid contrast set is a conjunction of attribute-value pairs that differ significantly in their distribution across groups. The search for valid contrast sets can produce a prohibitively large number of results which must be further filtered in order to be examined by a domain expert and have decisions enacted from them. In this paper, we introduce the notion of the minimum support ratio threshold to measure the ratio of maximum and minimum support across groups. We propose a contrast set mining technique to discover maximal valid contrast sets which meet a minimum support ratio threshold. We also introduce five interestingness measures and demonstrate how they can be used to rank contrast sets. Our experiments on real datasets demonstrate the efficiency and effectiveness of our approach, and the interestingness of the contrast sets discovered.**

Keywords-contrast set mining; group differences; data mining.

I. INTRODUCTION

Discovering the differences between groups is a fundamental problem in many disciplines. Groups are defined by a selected property that distinguish one group from the other. The search for group differences can be applied to a wide variety of objects such as patients, organizations, molecules, and even time-lines. The group differences sought are novel, implying that they are not obvious or intuitive, potentially useful, implying that they can aid in decision-making, and understandable, implying that they are presented in a format easily understood by human beings. It has previously been demonstrated that contrast set mining is an effective method for mining group differences from observational multivariate data [1] [2] [3] [4] [5].

Existing contrast set mining techniques can produce a prohibitively large set of differences across groups with varying levels of interestingness [2] [5]. For example, suppose we want to find out which demographic and socio-economic characteristics differentiate between women who use short-term, long-term, or no contraceptive methods. We could use data, as shown in Table I, with five such characteristics: wife currently working, husband currently working, has children, has high standard of living, and has high media exposure, where 1 indicates the characteristic is true, and 0 that it is false. There are 30 possible combinations of characteristics that differentiate between the women, however, they are not all equally interesting. For instance, assume we found that all the women who are working and have high media exposure use either short-term or long-term methods whereas those who are not working and do not have high media exposure, are equally likely to use either a short-term, long-term or no contraceptive method. Perhaps then, we could use the former result in a media marketing campaign targeted to that specific group of women, while the latter result which is less conclusive, is considered "uninteresting". We propose using a measure, called the minimum support ratio threshold, to discover "interesting" group differences during the search process.

Table I
SAMPLE DATASET

TID	wife currently working A	husband currently working B	has children C	has high standard of living D	has high media exposure E
1	1	0	1	1	1
2	0	1	1	0	1
3	1	1	0	0	1
4	0	0	1	1	1
5	1	1	1	1	1
...	...	...	...	...	...
768	0	0	0	1	1

The remainder of this paper is organized as follows. In Section II, we briefly review related work. In Section III, we describe the correlated contrast set mining problem. In Section IV, we provide an overview of the search framework for contrast set mining. In Section V, we introduce our algorithm for mining contrast sets that meet our minimum support ratio threshold. In Section VI, we present a summary of experimental results from a series of mining tasks. In Section VII, we conclude and suggest areas for future work.

II. RELATED WORK

The STUCCO (Search and Testing for Understandable Consistent Contrasts) algorithm [1] is the original technique for mining contrast sets. The objective of STUCCO is to find statistically significant contrast sets from grouped categorical data. It employs a modified Bonferroni statistic to limit Type

 ISBN: 978-1-61208-188-5

I errors resulting from multiple hypothesis tests. In other work, STUCCO forms the basis for a method proposed to discover negative contrast sets [6] that can include negation of terms in the contrast set. The main difference is their use of Holm's sequential rejective method [7] for the independence test.

The CIGAR (Contrasting Grouped Association Rules) algorithm [2] is a contrast set mining technique that specifically identifies which pairs of groups are significantly different and whether the attributes in a contrast set are correlated. CIGAR utilizes the same general approach as STUCCO, however it focuses on controlling Type II errors through increasing the significance level for the significance tests, and by not correcting for multiple comparisons. Like STUCCO, CIGAR is only applicable to discrete-valued data.

Contrast set mining has also been applied to continuous data. Early work focussed on the formal notion of a time series contrast set and an efficient algorithm was proposed to discover contrast sets in time series and multimedia data [8]. Another approach utilized a modified equal-width binning interval, where the approximate width of the intervals is provided as a parameter to the model [3]. The methodology used is similar to STUCCO except that the discretization step is added before enumerating the search space.

The COSINE (Contrast Set Exploration using Diffsets) algorithm [4] is a contrast set mining technique that uses a vertical data format, diffsets, a back tracking search algorithm, and simple discretization in mining maximal contrast sets from both discrete and continuous-valued attributes. The results demonstrate that COSINE is more efficient than STUCCO and CIGAR, even at very low minimum support difference thresholds. The GENCCS (Generate Correlated Contrast Sets) algorithm [5] is a contrast set mining technique that extends COSINE by utilizing mutual information and all-confidence to select the attribute-value pairs that are most highly correlated. The results show that GENCCS is more efficient and produced more interesting contrast sets than STUCCO and CIGAR.

III. Problem Definition

Let $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ be a set of n distinct attributes. We use $\mathcal{Q}$ and $\mathcal{C}$ to denote the set of *quantitative* attributes and the set of *categorical* attributes respectively. Let $\mathcal{V}(a_k)$ be the domain of values for a_k. An *attribute-interval pair*, denoted as $a_k : [v_{kl}, v_{kr}]$, is an attribute a_k associated with an interval $[v_{kl}, v_{kr}]$, where $a_k \in \mathcal{A}$ and $v_{kl}, v_{kr} \in \mathcal{V}(a_k)$. Further, if $a_k \in \mathcal{C}$ then $v_{kl} = v_{kr}$. Similarly, if $a_k \in \mathcal{Q}$, then $v_{kl} \leq v_{kr}$. Let $\mathcal{T} = \{x_1, x_2, \ldots, x_p\}$ where $x_k \in \mathcal{V}(a_k)$, $1 \leq k \leq p$, be a *transaction*. Let $\mathcal{D}$ be a set of transactions, called the *database*. Let $\{i_1, i_2, \ldots, i_m\}$ be a set of m distinct values from the set $\{1, 2, \ldots, n\}$, $m < n$. Let $\mathcal{F} = \{a_{i_1}, a_{i_2}, \ldots, a_{i_m}\}$, $a_{i_k} \in \mathcal{A}$, be a set of m distinct class attributes.

Let $G = \{a_1 : [v_{1l}, v_{1r}], \ldots, a_m : [v_{ml}, v_{mr}]\}$, $a_k \in \mathcal{F}, 1 \leq k \leq m$, $a_i \neq a_j$, $\forall i, j$, be a set of m distinct class attribute-interval pairs, called a *group*. Let $X = \{a_1 : [v_{1l}, v_{1r}], \ldots, a_q : [v_{ql}, v_{qr}]\}$, $a_i, a_j \in \mathcal{A} - \mathcal{F}$, $1 \leq k \leq q \leq n - m$, $a_i \neq a_j$, $\forall i, j$, be a set of distinct attribute-interval pairs, called a *quantitative contrast set*. Henceforth, we refer to a quantitative contrast set as simply a *contrast set*. A contrast set, X, is called k-specific, if $|X| = k$. The *support* of a contrast set, X, *in a database*, $\mathcal{D}$, denoted as $supp(X)$, is the percentage of transactions in $\mathcal{D}$ containing X. The support of a contrast set, X, in a group, G, denoted as $supp(X, G)$, is the percentage of transactions in $\mathcal{D}$ containing $X \cup G$.

A contrast set X associated with b mutually exclusive groups. $G_1, G_2, \ldots, G_b$ is called a *valid contrast set* (CS) if, and only if, the following four criteria are satisfied:

$$\exists ij supp(X, G_i) \neq supp(X, G_j), \tag{1}$$

$$\max_{ij} |supp(X, G_i) - supp(X, G_j)| \geq \epsilon, \tag{2}$$

$$supp(X) \geq \sigma, \tag{3}$$

and

$$\max_{i}^{n} \left\{ \frac{supp(Y, G_i)}{supp(X, G_i)} \right\} \geq \kappa, \tag{4}$$

where ϵ is called the *minimum support difference threshold*, σ is the *minimum frequency threshold*, κ called the *minimum subset support ratio threshold*, and $Y \subset X$ with $|Y| = |X| + 1$. Criterion 1 ensures that the contrast set represents a true difference between the groups. Contrast sets that meet this criterion are called *significant*. Criterion 2 ensures the effect size. Contrast sets that meet this criterion are called *large*. Criterion 3 ensures that the contrast set occurs in a large enough number of transactions. Contrast sets that meet this criterion are called *frequent*. Criterion 4 ensures that the support of the contrast set in each group is different from that of its superset. Contrast sets that meet this criterion are called *specific*.

A valid contrast set is called *maximal* if it is not a subset of any other valid contrast set. A valid contrast set is called *interesting* if the ratio of maximum and minimum support across the groups is sufficiently large. Formally, for a valid contrast set, X, the ratio is given by

$$\lambda(X) = \begin{cases} \infty, & \text{if } \min_{i}^{n} \{supp(X, G_i)\} = 0, \\ \dfrac{\max_{i}^{n} \{supp(X, G_i)\}}{\min_{i}^{n} \{supp(X, G_i)\}}, & \text{otherwise.} \end{cases} \tag{5}$$

A large value for $\lambda(X)$ implies that X occurs in significantly fewer transactions in one group G_i than in some other group

G_j. A value of ∞ indicates that X is absent from at least one group G_i and present in at least one other group G_j.

A valid contrast set is called a λ *contrast set* (λ-CS) if, and only if,

$$\lambda(X) \geq \omega, \tag{6}$$

where ω is a user-defined *minimum support ratio threshold*. This criterion ensures that the ratio of maximum and minimum support across all groups is sufficiently large. A λ contrast set is called a ∞ *contrast set* (∞-CS) if, and only if,

$$\lambda(X) = \infty. \tag{7}$$

Given a database $\mathcal{D}$, a minimum support difference threshold ϵ, a minimum frequency threshold σ, a minimum subset support ratio threshold κ, and a minimum support ratio threshold ω, our goal is to find all the maximal λ contrast sets (i.e., all maximal valid contrast sets that satisfy Equations 6 and 7).

IV. Background

A. Search for Contrast Sets

Our algorithm uses a backtracking search paradigm in order to enumerate all maximal group differences. Backtracking algorithms are useful because they allow us to iterate through all the possible configurations of the search space. Consider the partial search space tree shown in Figure 1. The root of the tree corresponds to the combine set $\{A:[0,0], A:[1,1], B:[0,0], B:[1,1], C:[0,0], C:[1,1], D:[0,0], D:[1,1], E:0,0], E:[1,1]\}$, which is composed of the 1-specific contrast sets from the attributes shown in Table I. Each attribute can take a value of 0 or 1. All these contrast sets share the empty prefix in common.

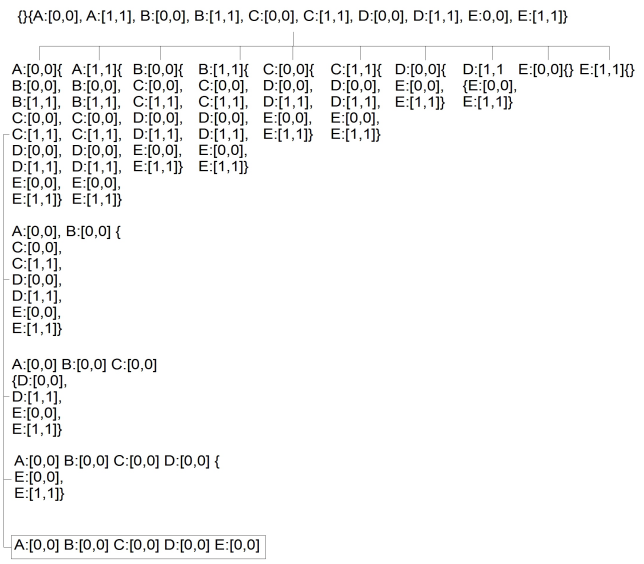


Figure 1. Search Tree: Box indicates a maximal contrast set

Formally, for a set of contrast sets with prefix P, $[P] = \{X_1, X_2, \cdots, X_n\}$, the intersection of PX_i with all of PX_j with $j > i$ is performed to obtain a new combine set $[PX_i]$ where the contrast set PX_iX_j satisfies Equations 1, 2, 3, 4, and 6. For example, from $[A:[0,0]] = \{B:[0,0], B:[1,1], C:[0,0], C:[1,1], D:[0,0], D:[1,1], E:[0,0], E:[1,1]\}$, we obtain $[A:[0,0]B:[0,0]] = \{C:[0,0], C:[1,1], D:[0,0], D:[1,1], E:[0,0], E:[1,1]\}$ for the next level of the search tree. A node with an empty combine set such as $[E:[0,0]$ need not be explored further.

B. Data Format

Contrast set mining algorithms using the vertical format have been shown to be very effective and usually outperform horizontal approaches [5] [4]. Our algorithm also uses a vertical data format in representing the data.

C. Ranking Methods

A contrast set mining task has the potential to return many contrast sets. Consequently, measures are needed to rank the relative interestingness of the contrast sets prior to presenting them to the end-user. Much work has been done on various measures of interestingness. For more on this, see [9] [10]. Ideally, a measure would be used to rank the contrast sets as well as describe them, akin to the support, confidence, leverage and lift measures used in association rule mining. In this section, we propose four measures and demonstrate their use in ranking contrast sets.

Here we define the variables used in the ranking methods described in this section. A contrast set, X, is represented by a set of association rules, $X \rightarrow G_1, X \rightarrow G_2, \ldots, X \rightarrow G_n$, where $G_1, G_2, \ldots, G_n$ are unique groups. Let $n(X, G_i)$ be the number of instances of X in G_i. Let $n(X, \neg G_i)$ be the number of instances of X in groups other than G_i (that is, the number of times X occurs in $G_1, \ldots, G_{i-1}, G_{i+1}, G_n$). Let $(\neg X, G_i)$ be the number of instances of contrast sets other than X in G_i. Let $n(\neg X, \neg G_i)$ be the number of instances of contrast sets other than X in groups other than G_i. Let N be the total number of instances.

The values $n(X, G_i), n(X, \neg G_i), n(\neg X, G_i)$, and $n(\neg X, \neg G_i)$ actually correspond to the observed frequencies at the intersection of the rows and columns in a 2×2 contingency table, such as the one shown in Table II for the association rule $X \rightarrow G_i$. Rows represent the occurrence of the contrast set and the columns represent occurrence of the groups.

Table II
CONTINGENCY TABLE FOR $X \rightarrow G_i$

	G_i	$\neg G_i$	Σ *Row*
X	$n(X, G_i)$	$N(X, \neg G_i)$	$n(X)$
$\neg X$	$n(\neg X, G_i)$	$n(\neg X, \neg G_i)$	$n(\neg X)$
Σ *Column*	$n(G_i)$	$n(\neg G_i)$	N

Distribution Difference: The *distribution difference* (DD) of a contrast set measures how different the support for a group in the contrast set is from the support for the group in the entire dataset [3]. Formally, the distribution difference for X, in G_i is given by

$$DD(X \rightarrow G_i) = \left| \frac{n(X, G_i)}{n(X)} \times \frac{N}{n(G_i)} - 1 \right|.$$

For example, assume that in the entire dataset 40% of individuals are male and 60% are female. Now assume that we have two contrast sets where 65% are male and 35% are female in the first, and 42% are male and 58% are female in the second. In comparing these contrast sets, the first is more interesting because it deviates more from the distribution in the entire dataset. The distribution difference captures that. The distribution difference can have a minimum value of zero, which indicates that the instances in the contrast set occur in the same distribution across the groups in comparison to the distribution in the entire dataset. A large distribution difference indicates significant variance in the distribution across the groups.

The *aggregate distribution difference* of a contrast set is the sum of the distribution difference values over all the groups. Formally, the aggregate distribution difference for X across $G_1, G_2, \ldots, G_n$ is given by

$$\textit{Aggregate } DD(X) = \sum_{i}^{n} DD(X \rightarrow G_i).$$

Unusualness: Unusualness is a measure of interestingness used in subgroup discovery [11]. Given a set of instances possessing some property of interest, a *subgroup* is a subset of instances in which the statistical characteristics of the property of interest are "unusual". Instances in the subgroup can be described by an association rule, $X \rightarrow Y$, where the property of interest is represented by the consequent, Y, and the antecedent, X, an itemset. Weighted relative accuracy (WRAcc) is used to evaluate the quality (i.e., unusualness) of the induced association rules.

Contrast set mining has been shown to be an extension of subgroup discovery, where each group represents a different property of interest [12]. Thus, we can use the weighted relative accuracy to measure the *unusualness* (UN) of X in G_i. Formally, unusualness is given by

$$\begin{aligned} UN(X, G_i) &= p(X) \times (p(G_i|X) - p(G_i)) \\ &\approx \frac{n(X)}{N} \times \left(\frac{n(X, G_i)}{n(X)} - \frac{n(G_i)}{N} \right). \end{aligned}$$

Possible values for unusualness range from -1 to 1. The unusualness of X, is determined by the group for which the unusualness is largest. Thus, the unusualness of X, across $G_1, G_2, \ldots, G_n$ is given by

$$\textit{Maximum } UN(X) = \max_{i} UN(X, G_i).$$

Coverage: The *coverage* of an association rule, $X \rightarrow G_i$ is the proportion of instances in the dataset where X is true in G_i [10], and is given by

$$\textit{Coverage}(X \rightarrow G_i) = p(X) = \frac{n(X)}{N}.$$

Possible values for coverage range from 0 to 1, inclusive, where contrast sets that occur more frequently have a higher coverage.

The coverage of a contrast set for all groups, called the *aggregate coverage*, is the sum of the individual coverage values for the contrast set in each group. The aggregate coverage of a contrast set X in $G_1, G_2, \ldots, G_n$ is given by

$$\textit{Aggregate Coverage}(X) = \sum_{i=1}^{n} \textit{Coverage}(X \rightarrow G_i)$$

Lift: The *lift* of an association rule, $X \rightarrow G_i$, measures how many times X and G_i actually occur together compared to the number of times X and G_i would be expected to occur together if they where statistically independent [13] and is given by

$$\textit{Lift}(X \rightarrow G_i) = \frac{p(X, G_i)}{P(X)P(G_i)} = \frac{N \times n(X, G_i)}{n(X) \times n(G_i)}$$

Possible values for lift range from 0 to infinity, inclusive.

The lift for a contrast set across all groups called the *aggregate lift*, is the sum of the lift for the contrast set in each group. Formally,

$$\textit{Aggregate Lift}(X) = \sum_{i=1}^{n} \textit{Lift}(X \rightarrow G_i).$$

Interestingness Factor: The four interestingness measures described above can be used individually to rank discovered contrast sets. However, they can can also be used in combination to determine the most interesting contrast sets based on multiple measures. The *Interestingness Factor* (IF) of a contrast set is the average of it's rank over a set of the selected interestingness measures, and is given by

$$IF(X) = \frac{\sum_{i=1}^{n} r_i \times w_i}{\sum_{i=1}^{n} w_i \times m_i}$$

where n is the number of interestingness measures, r_i is the rank of the contrast set by a measure i, w_i is the weight of the ranking measure i (i.e., a user-defined parameter indicating the relative importance of measure i), and m_i is the maximum rank of measure i. Possible values for the interestingness factor can range from 0 to 1, where values close to 0 indicate contrast sets which are more interesting, while values close to 1 indicate contrast sets which are less interesting. An interestingness factor of 1 indicates that the contrast set was ranked the lowest by each method.

 ISBN: 978-1-61208-188-5

V. MINING INTERESTING VALID CONTRAST SETS

GIVE (Generate Interesting Valid contrast sEts) presented in Algorithm 1, finds all the maximal valid contrast sets in a given dataset (i.e, all the contrast sets that satisfy Equations 1, 2, 3, 4, and 6). It adapts several tenets of the backtracking search technique proposed in [4] for contrast set mining.

Algorithm 1 GIVE($\mathcal{D}, \mathcal{F}, \epsilon, \sigma, \kappa, \omega, m$)

Input: A dataset, $\mathcal{D}$, and ranking method, m
Output: The set of ranked interesting valid contrast sets W

1: **for** each $i \in \mathcal{A}, \mathcal{A} \in \mathcal{D}, i \notin \mathcal{F}$ **do**
2: **if** $i \in \mathcal{Q}$ **then**
3: $\mathcal{V}(i) =$ DISCRETIZE(i)
4: **end if**
5: $B = B \cup \mathcal{V}(i)$
6: **end for**
7: $C_0 =$COMBINE$(\{\}, B, \epsilon, \sigma, 0, \omega, W)$
8: Sort each C_0 in increasing $|C_x|$ then in increasing F_x
9: TRAVERSE$(\{\}, C_0, W, \epsilon, \sigma, \kappa, \omega)$
10: RANK(W, m)
11: **return** W

GIVE begins by first determining all the 1-specific contrast sets from the domain $\mathcal{V}$ of each attribute in the dataset not occurring in $\mathcal{F}$, and storing them in B (lines 1 to 6). Quantitative attributes are discretized (line 3) to determine a $\mathcal{V}$ set from which 1-specific quantitative contrast sets can be generated. We use the discretization algorithm previously described in [4]. GIVE determines valid 1-specific contrast sets by calling the subroutine COMBINE, with the empty prefix $\{\}, B, \epsilon, \sigma, 0, \omega$, and W which will hold all the valid contrast sets at the end (line 7). The set of valid 1-specific contrast sets is re-ordered in increasing cardinality of the combine set and frequency (line 8). Thus contrast sets with a lower frequency at one level are less likely to produce contrast sets that meet our frequency threshold on the next level. GIVE then calls the subroutine, TRAVERSE, with the empty prefix, $\{\}, C_0, W, \epsilon, \sigma, \kappa,$ and ω (line 9). The valid contrast sets are ranked by a method m (line 10).

A. COMBINE

Given a prefix P, a combine set H, a set of valid contrast sets W, a minimum support difference threshold ϵ, a minimum frequency threshold σ, a minimum subset support ratio threshold κ, and a minimum support ratio ω, the COMBINE Algorithm, shown in Algorithm 3, builds new contrast sets from P and H that satisfy Equations 1, 2, 3, 4 and 6.

COMBINE begins by combining the prefix P with each member y of the possible set of combine elements, H, to create a new contrast set z (line 3). For each z, it calculates its diffset, D_z, its potential combine set, C_z, and its frequency, F_z (line 4). It then determines whether z satisfies Equations 1, 2, 3, and 4 (line 5). COMBINE also checks whether Equation 7 is satisfied (line 6). Any z which meets this criteria is potentially maximal and no further processing of the subset tree has to be done. z is added to W if it has no superset already in W (lines 7 to 9). Otherwise, COMBINE checks whether Equation 6 is satisfied. Any z which meets this criteria is added to the combine set, C (lines 10 to 11). Finally C is returned.

Algorithm 2 COMBINE($P, H, W, \epsilon, \sigma, \kappa, \omega$)

1: $C = \emptyset$
2: **for** each $y \in H$ **do**
3: $z = P \cup \{y\}$
4: Determine D_z, F_z, C_z, α_L
5: **if** significant(z, α_L) & large(z, ϵ) & frequent(z, σ) & specific(z, κ) **then**
6: **if** $\lambda(z) == \infty$ **then**
7: **if** $Z \not\supseteq z \cup H : Z \in W$ **then**
8: $W = W \cup \{z\}$
9: **end if**
10: **else if** $\lambda(z) \geq \omega$ **then**
11: $C = C \cup \{z\}$
12: **end if**
13: **end if**
14: **end for**
15: **return** C

B. TRAVERSE

Given a prefix P_l, a combine set C_l, a minimum support difference threshold ϵ, a minimum frequency threshold σ, a minimum subset support ratio threshold κ, and a minimum support ratio threshold ω, the TRAVERSE Algorithm, shown in Algorithm 3, traverses the search space for all, maximal or λ contrast sets that satisfy Equations 1, 2, 3, 4, and 6.

TRAVERSE begins by determining the next prefix, P_{l+1} (line 2). It then determines a new possible set of combine elements, H_{l+1}, by first stripping the prefix P_{l+1} of the previous prefix P_l, creating $P_{l+1}^{'}$ (line 4). It then determines from the list of elements in C_l, those which are greater than (appear after) P_{l+1} (recall from above, that P_{l+1} was also chosen from C_l) (line 6). For any such element, y, TRAVERSE strips it of the prefix P_l, creating $y^{'}$ (line 7). It then checks whether $P_{l+1}^{'}$ is not equal to $y^{'}$ and whether it is in the combine set of P_l (line 8). $P_{l+1}^{'}$ and $y^{'}$ are 1-specific contrast sets and if they originate from the same attribute, they cannot be part of a new contrast set, as we require contrast sets to have unique attributes. If $y^{'}$ is in the combine set of P_l then it will be in the combine set of P_{l+1}. If both conditions are met, y is added to H_{l+1} (line 9). TRAVERSE repeats this for every member of C_l.

 ISBN: 978-1-61208-188-5

Algorithm 3 TRAVERSE$(P_l, C_l, W_l, \epsilon, \sigma, \kappa, \omega)$

```
1:  for each x ∈ C_l do
2:     P_{l+1} = {x}
3:     H_{l+1} = ∅
4:     Let P'_{l+1} = P_{l+1} − P_l
5:     for each y ∈ C_l do
6:        if y > P_{l+1} then
7:           Let y' = y − P_l
8:           if y' ≠ P'_{l+1} & y' ∈ C_{P_l} then
9:              H_{l+1} = H_{l+1} ∪ {y}
10:          end if
11:       end if
12:    end for
13:    if |W_l| > 0 then
14:       if Z ⊇ P_{l+1} ∪ H_{l+1} : Z ∈ W_l then
15:          return
16:       end if
17:    end if
18:    C_{l+1} = COMBINE(P_{l+1}, H_{l+1}, W_l, ε, σ, κ, ω)
19:    Sort C_{l+1} by increasing F_z, ∀z ∈ C_{l+1}
20:    if C_{l+1} ≠ ∅ then
21:       if Z ⊉ P_{l+1} : Z ∈ W_l then
22:          W_l = W_l ∪ P_{l+1}
23:       end if
24:    else
25:       W_{l+1} = {W ∈ W_l : x ∈ W}
26:    end if
27:    if C_{l+1} ≠ ∅ then
28:       TRAVERSE(P_{l+1}, C_{l+1}, W_{l+1}, ε, σ, κ, ω)
29:    end if
30:    W_l = W_l ∪ W_{l+1}
31: end for
```

The cardinality of the current set of contrast sets, is determined and if it is greater than zero, TRAVERSE checks if $P_{l+1} \cup H_{l+1}$ is subsumed by an existing contrast set. If yes, the current and subsequent contrast sets in C_l can be pruned away (lines 13 to 17). If not, an extension is necessary. TRAVERSE then creates a new combine set for the next level of the search by using the subroutine COMBINE (line 18). The combine set, C_{l+1}, is sorted in increasing order of the frequency of its members (line 19). Any contrast set not in the combine set refers to a node pruned from the search tree. TRAVERSE checks if C_{l+1} is empty and if P_{l+1} is not a subset of any contrast set in W_l, P_{l+1} is added to W_l (lines 20 to 23). Otherwise, a new set of local contrast sets, W_{l+1}, is created based such that only the contrast sets in W_l that contain all the contrast sets in P_l are added to W_{l+1} (line 25). This allows the number of contrast sets of interest to be narrowed down as recursive calls are made. If C_{l+1} is not empty, TRAVERSE is called again with P_{l+1}, C_{l+1}, and the set of new local maximal contrast sets, W_{l+1} (lines 27 to 28). After the recursion completes, the set of maximal valid contrast sets or λ valid contrast sets, W_l, is updated with the elements from W_{l+1} (lines 30).

Table III
DATASET DESCRIPTION

Data Set	# Transactions	# Attributes	# Groups
Census	32561	14	5
Mushroom	8124	22	2
Spambase	4601	58	2
Waveform	5000	41	3

VI. EXPERIMENTAL RESULTS

In this section, we present the results of an experimental evaluation of our approach. Our experiments were conducted on an Intel dual core 2.40GHz processor with 4GB of memory, running Windows 7 64-bit. Discovery tasks were performed on four real datasets obtained from the UCI Machine Learning Repository [14]. Table III lists the name, number of transactions, number of attributes, and the number of groups for each dataset. These datasets were chosen because of their use with previous contrast set mining techniques and the ability to mine valid contrast sets with high specificity.

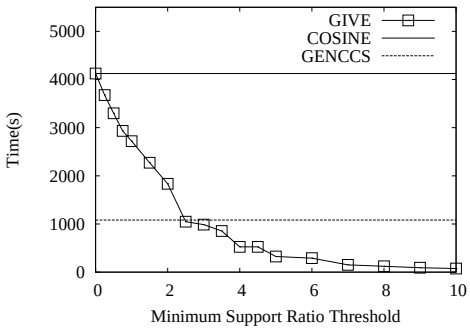
A. Performance of GIVE

We first examine the efficiency of GIVE by measuring the time taken to complete the contrast set mining task as the minimum support ratio threshold (MSRT) varies. We set the significance level to 0.95, the minimum support difference and minimum subset ratio to 0, respectively, and average the results over 10 consecutive runs. Figure 2 shows the number of valid contrast sets discovered and the run time for each of the datasets as the MSRT is varied. Results are only shown for MSRT values which produce substantial changes in the time. The time taken by COSINE and GENCCS for each dataset is also provided for comparison. For GENCCS, we set use the mean mutual information, and mean all confidence value, as the mutual information threshold, and all confidence threshold, respectively, as these were shown previously to be optimal [5]. Figures 2a, 2b, 2c, and 2d, show that GIVE is as efficient as COSINE when the minimum support ratio threshold is 0 but less than that of GENCCS. GIVE becomes more efficient than GENCCS when the MSRT is greater than 2.5, 0.75, 0.25, and 0.75 for the Spambase, Mushroom, Waveform and Census datasets, respectively. Since the MSRT serves as a constraint, as we increase its value, fewer contrast sets satisfy this constraint and GIVE becomes more efficient.

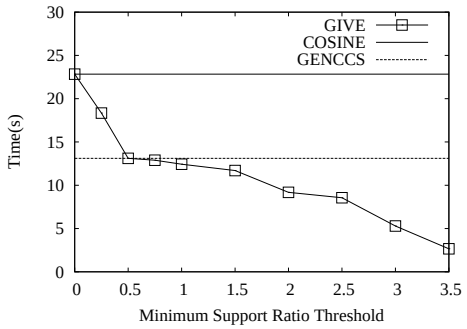
B. Effectiveness of GIVE

We examine the effectiveness of GIVE by measuring the average unusualness of the valid contrast sets discovered

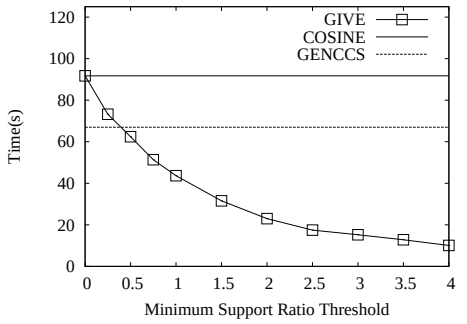
 ISBN: 978-1-61208-188-5



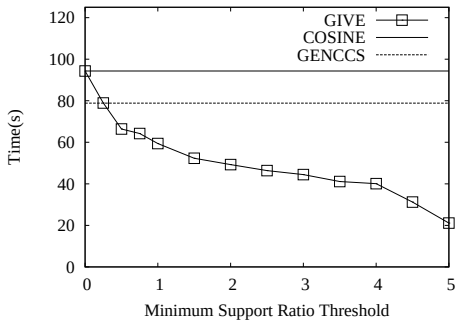
(a) SpamBase Runtime



(b) Mushroom Runtime

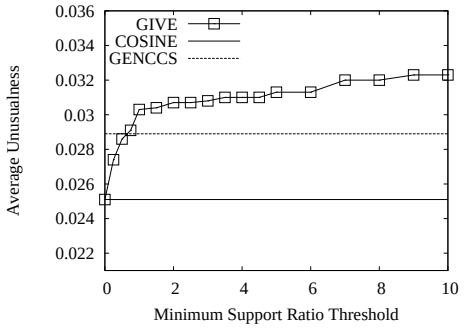


(c) Census Runtime

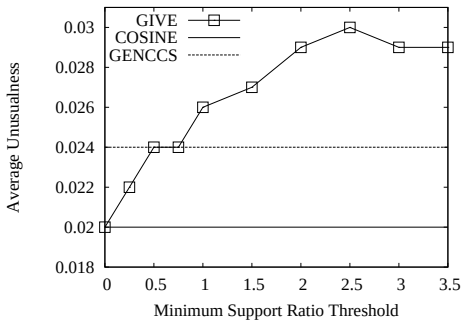


(d) Waveform Runtime

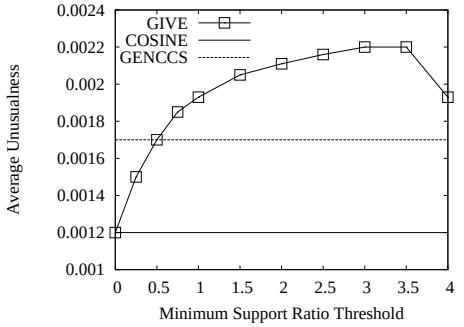
Figure 2. Summary of runtime results



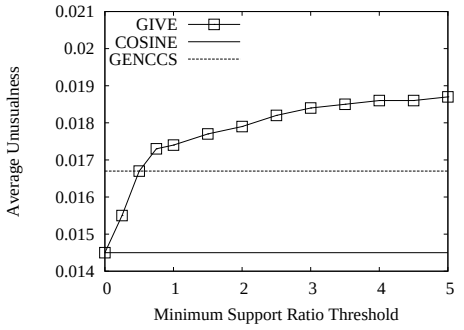
(a) Spambase Runtime



(b) Mushroom Runtime



(c) Census Runtime



(d) Waveform Runtime

Figure 3. Summary of interestingness results

as the MSRT varies, as shown in Figure 3. The average unusualness of the valid contrast sets discovered by COSINE and GENCCS for each dataset is also provided for comparison. Figure 3a, 3b, 3c, and 3d shows that the maximal contrast sets discovered by GIVE are more interesting, when measured by the average unusualness, than those discovered by either GENCCS or COSINE. The magnitude of the difference is significant even at lower

 ISBN: 978-1-61208-188-5

MSRT values where GENCCS outperforms GIVE as shown in Figures 2a, 2c, 2b, and 2d, which implies that even though GENCCS is less expensive, GIVE produces better quality contrast sets. Similar results were observed for the average distribution difference and lift, respectively, and are not shown due to space considerations.

C. Effect of λ on the Search Process

We further explored how the quality of the contrast sets discovered is affected by using λ in the search process by comparing the average interestingness factor for contrast sets that are found by COSINE and GENCCS that are also found by GIVE with those that are not found by GIVE. Table IV shows the average IF using all four measures equally weighted for each of the four datasets. In Table IV,

Table IV
EFFECTIVENESS OF λ

Data Set	COSINE & GIVE	COSINE & ¬GIVE	GENCCS & GIVE	GENCCS & ¬GIVE
Census	0.45	0.54	0.34	0.42
Mushroom	0.38	0.49	0.32	0.39
Spambase	0.45	0.58	0.40	0.49
Waveform	0.66	0.69	0.57	0.65

each row shows the average IF of the contrast sets found by COSINE and GIVE, COSINE and ¬GIVE, GENCCS and GIVE, and, GENCCS and ¬GIVE, respectively for each dataset. For example, for the Census dataset, the average interestingness factor of the contrast sets found by COSINE that are also found by GIVE is 0.45. For the Census, Mushroom, and Spambase datasets, the average IF of the contrast sets found by both COSINE and GIVE is significantly lower than those found by COSINE only. This shows that using λ in the search process does not compromise the quality of the contrast sets discovered. With the Waveform dataset, this difference is smaller and not as significant. For GENCCS, a similar observation can be made, though the difference in the average IF is significantly different for all four datasets.

VII. CONCLUSION

In this paper, we introduced the notion of the minimum support ratio threshold and proposed a contrast set mining technique, GIVE, for mining maximal valid contrast sets that meet a minimum support ratio threshold. We compared our approach with two previous contrast set mining approaches, COSINE and GENCCS, and found our approach to be comparable in terms of efficiency but more effective in generating interesting contrast sets. We also introduce five interestingness measures and demonstrated how they can be used to rank contrast sets. In addition, the results showed that the contrast sets discovered by GIVE had an average interestingness factor that was significantly higher than those produced only by COSINE or GENCCS. Future work will incorporate space reduction techniques with additional interestingness measures.

REFERENCES

[1] S. D. Bay and M. J. Pazzani, "Detecting group differences: Mining contrast sets," *Data Min. Knowl. Discov.*, vol. 5, no. 3, pp. 213–246, 2001.

[2] R. Hilderman and T. Peckham, "A statistically sound alternative approach to mining contrast sets," *Proceedings of the 4th Australasian Data Mining Conference (AusDM'05)*, pp. 157–172, Dec. 2005.

[3] M. Simeon and R. J. Hilderman, "Exploratory Quantitative Contrast Set Mining: A Discretization Approach," in *ICTAI (2)*, 2007, pp. 124–131.

[4] ——, "COSINE: A Vertical Group Difference Approach to Contrast Set Mining," in *Canadian Conference on AI*, 2011, pp. 359–371.

[5] ——, "GENCCS: A Correlated Group Difference Approach to Contrast Set Mining," in *MLDM*, 2011, pp. 140–154.

[6] T.-T. Wong and K.-L. Tseng, "Mining negative contrast sets from data with discrete attributes," *Expert Syst. Appl.*, vol. 29, no. 2, pp. 401–407, 2005.

[7] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian Journal of Statistics*, vol. 6, pp. 65–70, 1979.

[8] J. Lin and E. J. Keogh, "Group sax: Extending the notion of contrast sets to time series and multimedia data," in *PKDD*, 2006, pp. 284–296.

[9] R. J. Hilderman and H. J. Hamilton, *Knowledge Discovery and Measures of Interest*. Norwell, MA, USA: Kluwer Academic Publishers, 2001.

[10] L. Geng and H. J. Hamilton, "Interestingness measures for data mining: A survey," *ACM Comput. Surv.*, vol. 38, no. 3, p. 9, 2006.

[11] B. Kavsek and N. Lavrac, "Apriori-sd: Adapting association rule learning to subgroup discovery," *Applied Artificial Intelligence*, vol. 20, no. 7, pp. 543–583, 2006.

[12] P. Kralj, N. Lavrac, D. Gamberger, and A. Krstacic, "Contrast set mining for distinguishing between similar diseases," in *AIME*, 2007, pp. 109–118.

[13] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur, "Dynamic itemset counting and implication rules for market basket data," *SIGMOD Rec.*, vol. 26, no. 2, pp. 255–264, 1997.

[14] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

 ISBN: 978-1-61208-188-5

Efficient Color-Based Image Segmentation and Feature Classification for Image Processing in Embedded Systems

Alexander Jungmann, Bernd Kleinjohann, Lisa Kleinjohann
C-LAB
University of Paderborn, Germany
{global, bernd, lisa}@c-lab.de

Maarten Bieshaar
Faculty of Computer Science,
Electrical Engineering and Mathematics
University of Paderborn, Germany
maarten@mail.uni-paderborn.de

***Abstract*—In this paper, a color based feature extraction and classification approach for image processing in embedded systems in presented. The algorithms and data structures developed for this approach pay particular attention to reduce memory consumption and computation power of the entire image processing, since embedded systems usually impose strong restrictions regarding those resources. The feature extraction is realized in terms of an image segmentation algorithm. The criteria of homogeneity for merging pixels and regions is provided by the color classification mechanism, which incorporates appropriate methods for defining, representing and accessing subspaces in the working color space. By doing so, pixels and regions with color values that belong to the same color class can be merged. Furthermore, pixels with redundant color values that do not belong to any pre-defined color class can be completely discarded in order to minimize computational effort. Subsequently, the extracted regions are converted to a more convenient feature representation in terms of statistical moments up to and including second order. For evaluation, the whole image processing approach is applied to a mobile representative of embedded systems within the scope of a simple real-world scenario.**

Keywords-Image Segmentation; Feature Classification; Run-Length Encoding; Moments; Embedded Systems

I. INTRODUCTION

Image processing is a crucial service needed by many intelligent applications like driver assistance, surveillance or production in order to obtain and analyze visual information about the current situation. Usually this data and processing intensive task has strong timing requirements and is often realized on embedded systems. Although the performance capabilities of such embedded systems rise steadily, image processing approaches that stick appropriately to the prevalent restrictions regarding computational power and memory are required. Therefore many image processing solutions are realized on special hardware components such as Field Programmable Gate Arrays (FPGAs) or Digital Signal Processors (DSPs) in order to increase the processing speed. But this reduces the portability of the approach. In contrast, the approach presented in this paper can be applied to any embedded system that provides a Linux based operating system., independent of the underlying CPU instruction set.

The focus of this paper lies on the low level image processing phase, namely the combination of an efficient image segmentation algorithm with a simple but flexible feature classification mechanism. Algorithms and data structures were optimized with regard to computational effort and memory consumption. By image segmentation the original pixel-based data is drastically reduced to a feature-level representation, which requires significantly less memory on the one hand and is more convenient as well as less computational expensive for subsequent information processing steps on the other hand. As a convenient feature descriptor, which further reduces the visual data produced by the image segmentation, an optimized representation of features based on statistical moments is used. For feature classification a memory efficient color class representation together with computationally efficient access methods was developed, that nevertheless allows a representation of sophisticated color spaces made up of several subspaces. The feature classification is realized by two flexible heuristics. They are integrated into the segmentation process, in order to avoid further processing of pixels in the original image that can be classified as not relevant due to their color, thus reducing the computational effort even further.

A simple evaluation scenario with an embedded system - the miniature robot BeBot [1] - was developed to prove the performance of the presented approach. Within this scenario, the BeBot only relies on visual data in terms of images captured by its camera and processed by the image processing approach presented in this paper. By doing so, subsequent object detection and information processing steps for autonomous behavior are facilitated.

This paper is organized as follows. At first, the basic segmentation algorithm as well as a convenient feature representation are presented in Section II. Section III subsequently describes a flexible color-based feature classification mechanism. Afterwards, the real-world scenario, in which the whole image processing approach was successfully applied, is introduced in Section IV, followed by the associated results in Section V. Some existing approaches that are similar with respect to the basic concepts are briefly discussed in Section VI. Finally, the paper concludes with Section VII.

 ISBN: 978-1-61208-188-5

II. Feature Extraction by Image Segmentation

The fundamental idea of image segmentation is the identification of contiguous blocks of pixels (so-called *regions*) that are homogeneous with respect to a pre-defined criterion. Afterwards, those regions are abstracted into *features* that are compactly described in terms of position, size and shape. By doing so, the original pixel-based data is drastically reduced to a feature-level representation, which requires significantly less memory. Furthermore, features are more convenient as well as less computationally expensive for subsequent information processing steps. In fact, the overall challenge of image segmentation on embedded systems is to extract adequate features in order to reduce the memory consumption and computational effort for subsequent processing steps while simultaneously minimizing the computational effort of the actual segmentation process.

For that reason, our approach was developed to process an entire image only once, while simultaneously identifying and extracting regions, which are in turn compactly represented in order to reduce the memory consumption.

A. Region Representation

A convenient representation of a region during the segmentation process is crucial since it depends on the available memory and computational power. For that reason, our segmentation approach incorporates the *Run-Length Encoding* concept [2]. Sequences of adjacent pixels are encoded as runs, where a single run is compactly represented by means of only three integer values:

$$run_i = (x_i, y_i, l_i), \tag{1}$$

with x_i, y_i being the coordinates of the starting pixel and l_i denoting the number of adjacent pixels within the same row. Furthermore, a region R may consist of a set of n runs:

$$R = \{(x_1, y_1, l_1), (x_2, y_2, l_2), \ldots, (x_n, y_n, l_n)\} \tag{2}$$

While adding a pixel to a region is nothing but incrementing the length l_i of the related run run_i, merging two regions is realized by simply appending the sequence of runs of the first region to the sequence of runs of the second region.

B. Segmentation Algorithm

The entire image segmentation process is depicted in Algorithm 1. The main loop (lines 8-43) iterates row by row over the whole image, starting at the topmost row. The inner iteration loop (lines 9-42) processes each pixel within a row once, starting with the leftmost pixel.

After identifying a possibly existing left adjacent run run_{left} as well as its associated region R_{left} (lines 10-14), a heuristic H_i (cf. Section III-B) is applied in order to check if the current pixel is interesting at all (line 15). If so, a heuristic H_m (cf. Section III-B) is applied in order to check if the current pixel and the left adjacent region R_{left} can be merged (line 17). If heuristic H_m succeeds, the *region growing* step takes place by adding the current pixel to the left adjacent run run_{left} (length of run_{left} is incremented by value 1) and updating the associated region R_{left} with respect to its average color. Afterwards, the algorithm directly continues with the next pixel (line 19).

If either heuristic H_i or H_m fails, the left adjacent run

Algorithm 1 Segmentation Algorithm

```
 1: img = latest camera image
 2: regions = {∅}
 3: length_min = minimal length of a single run
 4: /* heuristic for deciding if a pixel is interesting at all */
 5: H_i = configure(interesting_heuristic)
 6: /* heuristic for deciding if pixels and/or regions can be merged */
 7: H_m = configure(merge_heuristic)
 8: for all row ∈ img do
 9:    for all pixel ∈ row do
10:       run_left = left_adjacent_run(pixel)
11:       if run_left not exists then
12:          goto line 35 /* start new run */
13:       end if
14:       R_left = region(run_left) /* run_left's associated region */
15:       if H_i(pixel) is TRUE then
16:          /* current pixel is interesting */
17:          if H_m(pixel, R_left) is TRUE then
18:             add(run_left, pixel) /* region growing */
19:             goto line 9 /* continue with next pixel */
20:          end if
21:       end if
22:       /* finalize a previously built left adjacent run */
23:       if length(run_left) < length_min then
24:          remove(regions, R_left)
25:       else
26:          regions_top = top_adjacent_regions(run_left)
27:          for all R_top ∈ regions_top do
28:             if H_m(R_left, R_top) is TRUE then
29:                /* region merging */
30:                merge(R_left, R_top)
31:                remove(regions, R_top)
32:             end if
33:          end for
34:       end if
35:       if H_i(pixel) is TRUE then
36:          /* start new run with current pixel as first pixel */
37:          run_new = new_run(pixel)
38:          /* start new region with current run as first run */
39:          R_new = new_region(run_new)
40:          add(regions, R_new)
41:       end if
42:    end for
43: end for
44: return regions
```

run_{left} is considered to be complete. If its length is smaller than the pre-defined minimal length value $length_{min}$ (line 23), run_{left} as well as its associated region R_{left} are discarded by removing R_{left} from the set of heretofore identified regions (line 24). Otherwise, the algorithm proceeds with its *region merging* mechanism. For this purpose, all regions bordering run run_{left} at the top are identified. Subsequently, another iteration loop tries to identify every region R_{top} within $regions_{top}$ that can be merged with the run_{left} associated region R_{left} (lines 27-33) by again applying heuristic H_m. If H_m succeeds for two regions R_{top} and R_{left}, the regions are merged by appending all runs of region R_{top} to R_{left} (by simply changing pointers). Furthermore, region R_{left} is updated with respect to its average color, whereas region R_{top} is completely discarded by removing it from the set of heretofore identified regions.

Independent of the result of the previous region merging mechanism, a new run run_{new} with the current pixel as its starting position as well as a new region R_{new} with run_{new} as its first run are allocated. But only, if the current pixel is interesting. Last but not least, the region R_{new} is added to the set of heretofore identified regions.

In fact, the arrangement of the several conditional clauses are optimized in the final implementation in order to minimize their redundant invocations. However, the depicted algorithmic structure was chosen to ease understanding of the algorithm's overall functionality.

C. Feature Descriptor

For high level object detection processes, a convenient feature descriptor, which further reduces the visual data in comparison to the run-based representation, is required. Furthermore, it should incorporate a model of uncertainty that takes stochastic errors such as sensor noise into account. For that reason, we decided for an implicit representation in terms of statistical quantities.

By interpreting a region and its associated pixels as a two-dimensional Gaussian distribution in the image plane, a region can be implicitly described by means of statistical parameters, namely the two mean values m_x and m_y, the two variances σ_x^2 and σ_y^2, and the covariance σ_{xy}. A generalization of these specific parameters are the statistical moments. In this context, the two mean values correspond to the two moments of first order (m_{10} and m_{01}), whereas the two variances and the covariance correspond to the centralized (or central) moments of second order (μ_{20}, μ_{02} and μ_{11}):

$$\vec{m} = \begin{pmatrix} m_x \\ m_y \end{pmatrix} = \begin{pmatrix} m_{10} \\ m_{01} \end{pmatrix} \tag{3}$$

$$\Sigma = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{pmatrix} = \begin{pmatrix} \mu_{20} & \mu_{11} \\ \mu_{11} & \mu_{02} \end{pmatrix} \tag{4}$$

In the field of digital image processing, this statistical approach can be directly applied in terms of discretized moments [3]. In general, a discretized two-dimensional moment m_{pq} of order $p+q$ belonging to a region R is defined as

$$m_{pq} = \sum_{(x,y)\in R} x^p y^q, \tag{5}$$

meaning nothing but only the coordinates (x,y) that belong to R have to be considered for computing the sum. Furthermore, in our optimized approach, the required central moments of second order are efficiently computed by means of the moments up to and including second order:

$$\begin{aligned} \mu_{20} &= m_{20} - \frac{m_{10}^2}{m_{00}}, \\ \mu_{02} &= m_{02} - \frac{m_{01}^2}{m_{00}}, \\ \mu_{11} &= m_{11} - \frac{m_{10}\cdot m_{01}}{m_{00}}. \end{aligned} \tag{6}$$

Consequently, for representing a single region as a two-dimensional Gaussian distribution, our basic feature descriptor only comprises the following set M of moments:

$$\begin{aligned} M &= \{m_{pq} | p+q \leq 2\} \\ &= \{m_{00}, m_{10}, m_{01}, m_{11}, m_{20}, m_{02}\} \end{aligned} \tag{7}$$

For efficiently converting the region representation in terms of runs into the feature descriptor, the *Delta "δ" Method* [4] is applied by directly incorporating the numerical quantities of a run, namely the coordinates x_i and y_i of its starting pixel as well as its length l_i. For this purpose, let $S1_i$ and $S2_i$ be defined as follows:

$$\begin{aligned} S1_i &= \sum_{k=0}^{l_i-1} k = \frac{(l_i^2 - l_i)}{2} \\ S2_i &= \sum_{k=0}^{l_i-1} k^2 = \frac{l_i^3}{3} - \frac{l_i^2}{2} + \frac{l_i}{6} \end{aligned} \tag{8}$$

The required geometric moments m_{00_i}, m_{10_i}, m_{01_i}, m_{11_i}, m_{20_i} and m_{02_i} of a single run run_i are then computed in the following optimized way:

$$\begin{aligned} m_{00_i} &= l_i \\ m_{01_i} &= l_i \cdot y_i \\ m_{10_i} &= l_i \cdot x_i + S1_i \\ m_{02_i} &= l_i \cdot y_i^2 \\ m_{20_i} &= l_i \cdot x_i^2 + 2 \cdot S1_i \cdot x_i + S2_i \\ m_{11_i} &= l_i \cdot [l_i \cdot x_i + S1_i] = y_i \cdot m_{10_i} \end{aligned} \tag{9}$$

Finally, the moments of a region R correspond to the sums of the particular moments of all related n runs:

$$M = \{m_{pq} | m_{pq} = \sum_{i=1}^{n} m_{pq_i} \wedge p+q \leq 2\} \tag{10}$$

As shown by Prokop and Reeves [5], an adequate set of additional attributes can be derived from these moments

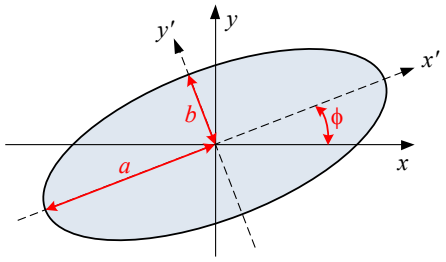


Figure 1. Representation of an ellipse by its major axis x', minor axis y' and angle of inclination ϕ.

(and central moments), in order to additionally describe a feature in a more explicit manner. In this context, the *mass* of a feature corresponds to the number of related pixels. It is, therefore, equivalent to the zeroth order moment m_{00}. Furthermore, the coordinates $(\bar{x},\bar{y})$ of the *center of mass* of a feature in the image plane are defined by means of the moments of zeroth and first order:

$$\bar{x} = \frac{m_{10}}{m_{00}} \quad \text{and} \quad \bar{y} = \frac{m_{01}}{m_{00}} \tag{11}$$

In addition, according to [6], by statistically describing a feature in terms of moments up to and including second order, the feature is equivalent to a *elliptical disk* (see Figure 1) with constant intensity, having definite size, orientation and eccentricity and being centered at the origin of the image plane. Its major radius a and minor radius b as well as its orientation ϕ (see Table I) can be in turn derived from the central moments in the following way:

$$a = \sqrt{\frac{2\left(\mu_{20}+\mu_{02}+\sqrt{(\mu_{20}-\mu_{02})^2+4\mu_{11}^2}\right)}{\mu_{00}}} \tag{12}$$

$$b = \sqrt{(\frac{2\left(\mu_{20}+\mu_{02}-\sqrt{(\mu_{20}-\mu_{02})^2+4\mu_{11}^2}\right)}{\mu_{00}}} \tag{13}$$

III. Feature Classification

The classification mechanism enables the user to define so called color classes for classifying pixels, thus features depend on their color. These color classes provide the basics for the heuristics H_i and H_m needed by the segmentation algorithm (cf. Section II-B).

A. Representation and Definition of the Color Classes

Color Classes in our context are nothing but an id representing a subspace within the working color space (i.e., YUV or RGB), which is in turn represented in terms of a cube (see Figure 2) with the particular color channels corresponding to the three dimensions. Each channel has a maximal resolution of 8bit, leading to $256\times256\times256$ possible combinations for addressing a position in the respective

Table I
Determining the angle of inclination ϕ as a function of the central moments of second order.

$\mu_{20}-\mu_{02}$	μ_{11}	ϕ	
0	0	0	
0	+	$+\pi/4$	$\xi = \frac{2\mu_{11}}{\mu_{20}-\mu_{02}}$
0	−	$-\pi/4$	
+	0	0	
−	0	$-\pi/2$	
+	+	$(1/2)\cdot\arctan\xi$	$(0<\phi<\pi/4)$
+	−	$(1/2)\cdot\arctan\xi$	$(-\pi/4<\phi<0)$
−	+	$(1/2)\cdot\arctan\xi+\pi/2$	$(\pi/4<\phi<\pi/2)$
−	−	$(1/2)\cdot\arctan\xi-\pi/2$	$(-\pi/2<\phi<-\pi/4)$

space. By realizing the color space as a look up table in terms of a one-dimensional array of integer values instead of a three-dimensional one, the number of memory accesses are minimized. For indicating, whether a coordinate within the color space is assigned to a color class with id i, the ith bit of the respective integer value is set to 1. Consequently, the number of possible color classes is bound to the chosen integer size (e.g., an 8bit integer allows the application of 8 different color classes). Additionally, by following this bit based approach, a coordinate can be simultaneously assigned to more than one color class at the same time. Furthermore, necessary mechanisms such as *inserting* and *removing* a color class as well as *testing* if a color tuple is assigned to a color class can be consequently implemented in terms of simple but very efficient bit operations.

The resolution of the channels can be restricted by defining a quantization factor $\lambda \in \{2^i | i = 1,2,\ldots,8\}$ in order to drastically reduce the over all memory consumption. Graphically, this means that the whole color space is no longer represented by $256\times256\times256$ different color combinations, but $256/\lambda\times256/\lambda\times256/\lambda$ statically allocated sub-cubes, where color combinations that are located within the same sub-cube are not further differentiated but considered as identical. A color class, thus in general is defined by a set of sub-cubes of size $\lambda\times\lambda\times\lambda$ in the color space, where sub-cubes may belong to several color classes as already men-

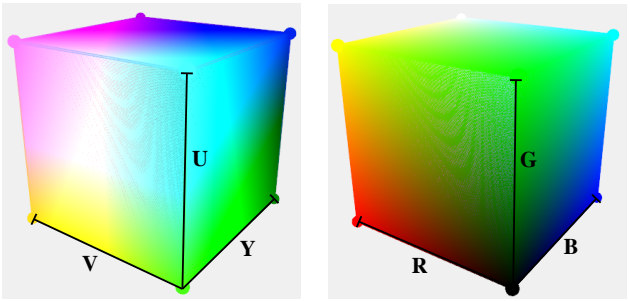


Figure 2. YUV and RGB color spaces, represented as three-dimensional cubes.

tioned above. As a consequence, the resolution and hence the granularity of the color space is significantly decreased, while the robustness of the overall feature classification is increased with respect to image noise. Dependent on the chosen quantization factor λ, the reduced granularity still leaves enough space for adequately distinguishing different colors.

Accessing the integer value of a color tuple (c_1, c_2, c_3) in the look up table *lut* is done by

$$lut[f(c_1, c_2, c_3)] \tag{14}$$

with

$$\begin{aligned} f(c_1, c_2, c_3) =& (c_1/\lambda) \cdot (256/\lambda)^2 \\ &+ (c_2/\lambda) \cdot 256/\lambda \\ &+ (c_3/\lambda) \end{aligned} \tag{15}$$

By only allowing powers of two as values for λ, the devision by λ can be replaced by a bit shift mechanism:

$$\begin{aligned} f(c_1, c_2, c_3) =& (c_1 \gg \lambda_s) \cdot (256 \gg \lambda_s)^2 \\ &+ (c_2 \gg \lambda_s) \cdot (256 \gg \lambda_s) \\ &+ (c_3 \gg \lambda_s), \end{aligned} \tag{16}$$

with

$$\lambda_s = log_2 \lambda. \tag{17}$$

The two multiplications can be also replaced as follows:

$$\begin{aligned} f(c_1, c_2, c_3) =& ((c_1 \gg \lambda_s) \ll c_{1s}) \\ &+ ((c_2 \gg \lambda_s) \ll c_{2s}) \\ &+ (c_3 \gg \lambda_s), \end{aligned} \tag{18}$$

with

$$c_{1s} = log_2((256/\lambda)^2) = log_2((256 \gg \lambda_s)^2) \tag{19}$$

and

$$c_{2s} = log_2(256/\lambda) = log_2(256 \gg \lambda_s). \tag{20}$$

Since λ_s, c_{1s} and c_{2s} have to be calculated only once whenever the quantization factor λ is changed (and hence the overall resolution of the color space), calculating the position as a function of a color tuple (c_1, c_2, c_3) for accessing the appropriate integer value in the look up table is very efficient.

In order to define a color class i as a subspace within the color space, the well known *Flood fill* [7] approach is applied. In this context, a user has to select a particular pixel in order to indicate the starting position in a camera image. Dependent on the configuration parameters, the algorithm recursively detects pixels with similar color in the adjacent neighborhood, taking each detected pixel as a new starting position. After termination, the color tuples (c_1, c_2, c_3) of all identified pixels are marked within the color space to be associated to color class i by setting bit i of the related sub-cube (i.e. of the integer value at the particular position in the one-dimensional array) to 1.

B. Heuristics for Segmentation

The necessary heuristics H_m for deciding whether a region growing or region merging step should take place and H_i for determining if a given pixel is interesting at all can now be easily constructed.

Let (c_1, c_2, c_3) be the color values of the current pixel. By applying (18), the associated position in the previously mentioned look up table can be computed and accessed. Then, by simply checking whether the value read from the memory is different to zero or not (i.e., if at least one bit is set), the heuristic H_i is able to determine, if the pixel is interesting or not:

$$\begin{aligned} H_i(pixel) &= H_i(c_1, c_2, c_3) \\ &= \begin{cases} 1 & if \quad lut[f(c1, c2, c3)] \neq 0 \\ 0 & else. \end{cases} \end{aligned} \tag{21}$$

For allowing that, e.g., two regions can be merged, their estimated average color values (c_1, c_2, c_3) and (c_4, c_5, c_6) have to be assigned to the same color class. For that purpose, the respective entries in the look up table have to be read and compared. Since the assignment of a color class in the look up table is done by activating bits, the comparison of the two integer values can be simply implemented by a bitwise *AND* operator (&) and a subsequent test for 0, meaning that there are no bits, which are set to 1 in both values. Hence, heuristic H_m can be realized as follows:

$$\begin{aligned} H_m &= H_m((c_1, c_2, c_3), (c_4, c_5, c_6)) \\ &= \begin{cases} 1 & if \quad \begin{array}{c} (lut[f(c1, c2, c3)] \\ \& \\ lut[f(c4, c5, c6)]) \neq 0, \end{array} \\ 0 & else. \end{cases} \end{aligned} \tag{22}$$

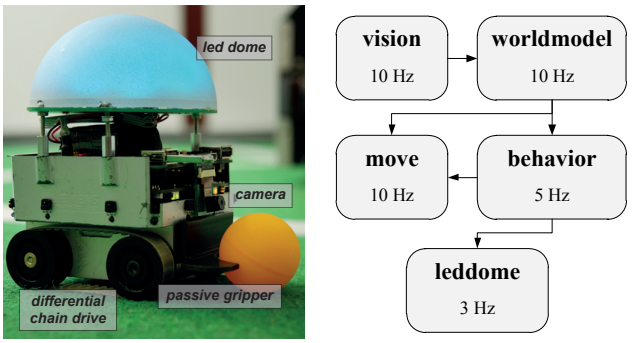
IV. Evaluation Scenario

For evaluating the functional principle of the presented approach, a simple real-world scenario that incorporates the miniature robot BeBot [1] was implemented.

A. BeBot Specifications

The BeBot is a miniature chain driven robot (shown in Figure 3(a)), which has been developed at the Heinz Nixdorf Institute of the University of Paderborn. Despite its small size (approximately 9 cm x 9 cm x 8 cm), it is equipped with modern technology distributed on two modular boards. The lower one of these boards implements basic functionalities such as controlling the motors by means of an ARM 7 based microcontroller. In contrast, the upper board constitutes a more powerful computing platform. Whereas a camera that is mounted at the front allows the BeBot to perceive its environment within a limited field of view, algorithms for convenient information processing steps can be executed in an embedded Linux environment running on an ARM Cortex-A8 CPU with a maximum frequency of 600 MHz

 ISBN: 978-1-61208-188-5



(a) (b)

Figure 3. (a) The miniature robot BeBot, pushing an orange ping-pong ball. (b) The node structure of the implemented behavior system.

accessing up to 256MB RAM of main memory. The BeBot is additionally equipped with a passive gripper, which enables the robot to push small objects such as ping-pong balls. Furthermore, by illuminating its so called led dome in arbitrary colors during runtime, the miniature robot can express its current state to human observers and other robots, respectively.

The network based TCP/IP communication is exclusively based on Bluetooth. For that reason, an approach that relies on sending raw visual data to external systems in order to apply computational expensive image processing algorithms is not feasible.

B. Scenario Description

The overall task of a BeBot in our evaluation scenario is to push a single-colored ping-pong ball through a slalom course, which consists of small pylons that are arranged in a straight line (see Figure 4 for a schematic illustration). For facilitating the identification of the start and the finish of the course, simple markers with a single color are attached to the respective points. Furthermore, the ping-pong ball is initially positioned somewhere around the course.

First of all, the robot has to search and catch the ball and return with it to the start point. Afterwards, it has to push the ball through the slalom course without loosing it. If the BeBot looses the ball, it has to start all over again. When successfully finished its drive through the course, the robot has to stop while blinking with its led dome.

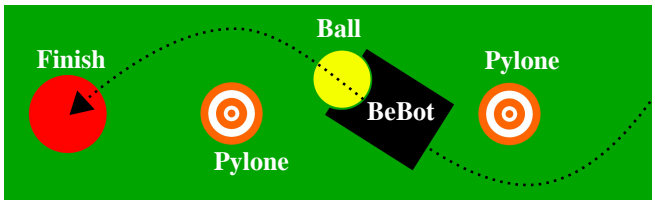


Figure 4. Schematic bird's eye view of the evaluation scenario.

C. Realization of the BeBot Behavior System

The node-based implementation of the BeBot's behavior system is depicted in Figure 3(b). The *vision* node incorporates all image processing steps including the presented segmentation and classification approach as well as a simple object detection mechanism in order to recognize the scenario's particular objects. The reduced visual information is subsequently transmitted to the *worldmodel* node, in which a very abstract representation of the BeBot's state within the environment is accumulated. The overall strategy is computed in the *behavior* node on a very abstract level. Those abstract behaviors (e.g., *DriveSlalom*, *CatchBall*) do not change quickly and do not contain any detailed information about how to achieve the respective goal. Furthermore, the connected *leddome* node changes the color of the led dome according to the current overall behavior. Finally, the low level behaviors (e.g., *MoveCurve*, *SearchBall*) that output the commands for the differential chain drive are implemented in the *move* node.

V. Results

The presented approach was completely implemented in C++ and was successfully applied to the BeBot within the scope of the evaluation scenario. The resolution of the images grabbed by the BeBot camera in the YUV color space was set to 320×240 pixels. Furthermore, the quantization factor λ was set to 16, resulting in a resolution of each channel in the YUV color space of $256/\lambda = 16$. Due to the setup of the scenario, the number of color classes were restricted to less then 8. For that reason, the look up table was initialized with 8bit integer values. Consequently, the overall memory consumption of the basic classification array was

$$(256/16 \cdot 256/16 \cdot 256/16) \cdot 1\,Byte = 4096\,Byte = 4\,kB \quad (23)$$

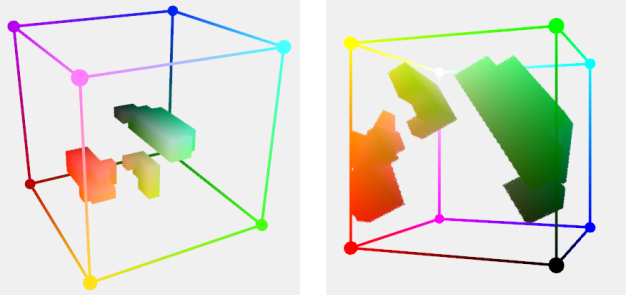
The subspaces corresponding to the different color classes within the working color space are depicted in Figure 5.

In order to demonstrate the efficiency of the introduced approach, a typical subjective view of the scenario setup (see Figure 6(a)) was selected for further experimental investigations. Within the scope of these experiments, the frequency of the upper board's ARM CPU was fixed to 520Mhz. Furthermore, the behavior system was disabled in order to run the image processing as standalone process.

Four different test cases were considered. The particular results are shown in Table II. In this context, case *A* solely contains the image acquisition step as evaluation basis. In addition, case *A* was used for identifying the maximum possible frame rate (16 fps) currently supported by the BeBot.

Case *B* incorporates a modified version of the presented segmentation approach: the classification mechanisms were completely discarded. Consequently, all pixels were supposed to be of interest and therefore considered during the

 ISBN: 978-1-61208-188-5



(a) YUV (b) RGB

Figure 5. (a) Representation of the different color classes (pylones, ball, marker) in the YUV color space, in which the whole image processing takes place. The influence of the quantization factor λ can be seen in terms of the topology of the particular subspaces, which consist of small cubes. Additionally, (b) depicts the defined color classes transformed into the RGB color space.

segmentation step. Furthermore, the original heuristic H_m (22) for deciding whether a region growing or region merging step should take place was replaced by a heuristic, which determines, whether two color tuples reside in a predefined neighborhood within the YUV color space. Figure 6(b) shows the processed image as well as the extracted features in terms of their respective centers of mass and ellipses.

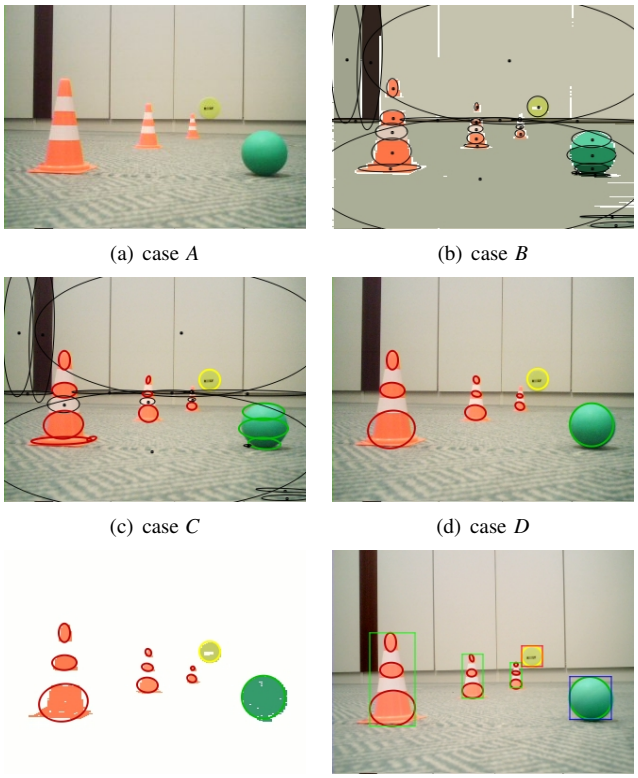
Based on the classification independent segmentation in case *B*, case *C* additionally incorporates the presented classification approach. However, the classification was not applied during the segmentation process on pixel level, but as a subsequent processing step on feature level. The classified features can be seen in Figure 6(c) in terms of their colored ellipses, where the color of an ellipse represents the related color class.

The advantages of the original approach (case *D*), where the classification heuristics H_i and H_m are applied during segmentation as described in Algorithm II-B, become obvious by comparing the performance values as well as the results that are depicted in Figure 6(d) and Figure 6(e) with the previously mentioned results. Since all redundant pixels that are not of interest are already discarded during the segmentation step, the amount of visual data is drastically reduced and only scenario relevant features are extracted. As a consequence, the workload of the CPU is significantly decreased. Furthermore, since only pixels that belong to the same color class are merged, regions being more proper are constructed during the segmentation step (compare, e.g., the three extracted and classified features of the ball in Figure 6(c) with the one feature of the ball in Figure 6(d)).

Table II
PERFORMANCE VALUES OF THE FOUR DIFFERENT TEST CASES.

Case	Related figures	CPU	Mem.	P. time	#F	fps
A	Fig. 6(a)	4%	8.6%	-	-	16
B	Fig. 6(b)	79%	10.3%	46.7 *ms*	29	16
C	Fig. 6(c)	80%	10.5%	47.1 *ms*	29	16
D	Fig. 6(d), Fig. 6(e)	29%	10.5%	15.3 *ms*	10	16

P. time: Average processing time (ms).
#F: Average number of extracted features.
fps: Frames per second.



(a) case *A* (b) case *B*

(c) case *C* (d) case *D*

(e) case *D* (f)

Figure 6. A BeBot's typical subjective view of the scenario setup: (a) original image, (b) result of image segmentation without incorporating classification, (c) subsequent feature classification, (d) original and (e) result image of the combined approach, (f) detected scenario relevant objects.

Finally, Figure 6(f) shows a representative result of the entire image processing (presented approach + object detection) within the scope of the evaluation scenario. While the classified features are again represented by means of their colored ellipses, the identified scenario relevant objects are represented by means of their bounding boxes.

VI. RELATED WORK AND DISCUSSION

A similar color-based segmentation and classification approach was presented in [8]. However, in their context, an image has to be processed in several steps. After an optional step of color space projection, each pixel is classified by one of up to 32 color classes. In comparison to our classification mechanism, the associated subspaces within the working color space can only take shape of convex cuboids, whereas our subspaces may also take shape of concave structures or either may consist of two and more unconnected subspaces.

Afterwards, each row is processed again in order to encode adjacent pixels of the same color class into runs.

 ISBN: 978-1-61208-188-5

Finally, connected runs belonging to the same color class are efficiently grouped into regions by means of a tree-based union find algorithm, whereas the constructed regions are represented in terms of their bounding boxes, centroids and sizes. Although the basic concepts are similar to our approach, there are significant differences with respect to the number of processing steps, the flexibility in representing color classes, the method and point of time of grouping runs as well as the robustness of the actual feature descriptor. In the latter case, especially the bounding boxes are error prone and cannot appropriately compensate the negative effects of stochastic errors such as image noise.

Another quite similar approach was presented in [9]. Whereas the classification mechanism is identical to the previously discussed one, the number of image processing steps are reduced to two. At first, similar to our approach, the image is compressed by comparing adjacent pixels with respect to their associated color classes and by encoding each contiguous block of pixels into a run. After completing a run, neighboring runs belonging to the same color class are identified and marked as connected. However, in comparison to our approach, they are not yet merged. For increasing the performance of the algorithm, pixels that are not of interest are discarded, just as our segmentation algorithm does. In the second step, a recursive depth-first traversal is performed in order to identify runs that were marked as connected and to assign them to the same feature, which is in turn represented by its center of gravity efficiently derived from the runs.

A fast component labeling and description algorithm that also bases on run length was introduced in [10]. Similar to our approach, the main idea is to scan an image from left to right row by row starting at the topmost row in order to construct runs of connected pixels. However, only binary thus already segmented images are considered. Consequently, in comparison to our approach, the entire image has to be processed at least twice in order to identify and extract regions. Furthermore, again, no robust feature representation in terms of statistical moments is derived from the grouped runs, but only the area, the center and the bounding box.

VII. Conclusion

In this paper, an efficient color-based image segmentation and feature classification approach was presented. The entire image has to be processed only once in order to identify regions by means of an efficient classification mechanism. During the segmentation process, regions are compactly represented in terms of runs in order to drastically reduce the overall memory consumption. Furthermore, the computational effort for constructing and merging regions is significantly reduced. The classification approach additionally provides an efficient mechanism for deciding whether a pixel is relevant for the segmentation process or not. As a consequence, redundant visual data is already discarded on pixel level. A second and final step efficiently converts the extracted regions into a convenient and robust feature representation in terms of statistical moments by directly incorporating the previously assembled runs. The presented results show the functional correctness of our combined approach as well as its efficiency in comparison to a sequential one. The presented approach constitutes a portable and flexible solution for any Linux based embedded system in order to overcome the data intensive and computationally extremely expensive task of image processing.

Acknowledgment

This work was partially supported by the German Research Foundation (DFG) within the Collaborative Research Center "On-The-Fly Computing" (SFB 901).

References

[1] S. Herbrechtsmeier, U. Witkowski, and U. Rückert, "Bebot: A modular mobile miniature robot platform supporting hardware reconfiguration and multi-standard communication," in *Progress in Robotics*. Springer Berlin Heidelberg, 2009, vol. 44, pp. 346–356.

[2] R. Jain, R. Kasturi, and B. G. Schunck, *Machine vision*. McGraw-Hill, 1995, (verified: 10.01.2012). [Online]. Available: http://www.cse.usf.edu/~r1k/MachineVisionBook/MachineVision.htm

[3] M.-K. Hu, "Visual pattern recognition by moment invariants," *IEEE Transactions on Information Theory*, vol. 8, no. 2, pp. 179–187, 1962.

[4] M. F. Zakaria, L. J. Vroomen, P. J. A. Zsombor-Murray, and J. M. H. M. van Kessel, "Fast algorithm for the computation of moment invariants," *Pattern Recogn.*, vol. 20, no. 6, pp. 639–643, 1987.

[5] R. J. Prokop and A. P. Reeves, "A survey of moment-based techniques for unoccluded object representation and recognition," *CVGIP: Graph. Models Image Process.*, vol. 54, no. 5, pp. 438–460, 1992.

[6] M. R. Teague, "Image analysis via the general theory of moments," *Journal of the Optical Society of America (1917-1983)*, vol. 70, pp. 920–930, 1980.

[7] L. Vandevenne, "Lode's computer graphics tutorial - flood fill," (verified: 10.01.2012). [Online]. Available: http://lodev.org/cgtutor/floodfill.html

[8] J. Bruce, T. Balch, and M. Veloso, "Fast and inexpensive color image segmentation for interactive robots," in *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2000, pp. 2061–2066.

[9] C. Messom, S. Demidenko, K. Subramaniam, and G. Gupta, "Size/position identification in real-time image processing using run length encoding," in *Proceedings of the 19th IEEE Instrumentation and Measurement Technology Conference (IMTC)*, 2002, pp. 1055–1059.

[10] L. Qiu and Z. Li, "A fast component labeling and description algorithm for robocup middle-size league," in *7th World Congress on Intelligent Control and Automation (WCICA)*, 2008, pp. 6575–6579.

 ISBN: 978-1-61208-188-5