Contents

Foreword ........................................... 5
Organizing and program committees ................. 8

Invited talks
Computational Intelligence in Vaasa .................. 9
Jarmo T. Alander

From an artificial intelligence to multiple subjective intelligences:
Approaching the second level of subjectivity .......... 11
Timo Honkela

Machine learning for abstraction in robotic systems .... 13
Ville Kyrki

Machine learning and language technology for the bio-health domain .... 14
Tapio Salakoski

From neural networks to artificial intelligence .......... 15
Harri Valpola

Tutorial
Multilingual document mining for unstructured information .......... 16
Jolanta Mizera-Pietraszko

Papers
Gaze contingent adaptive interface for tele-operation .......... 18
Zaheer Ahmed, Aamir Shahzad and Craig A. Lindley

Comparison of variational Bayes and Gibbs sampling in reconstruction of missing values with probabilistic principal component analysis ........ 25
Luis Gabriel De Alba Rivera, Alexander Ilin and Tapani Raiko

Experiments with domain adaptation methods for statistical MT:
From European parliament proceedings to Finnish newspaper text ........ 31
Marcus Dobrinkat and Jaakko J. Väyrynen

GArphics — Applying genetic algorithm for generating graphics .......... 39
Anssi Jäntti and Jarmo T. Alander

A synchronized system concept and a reference implementation for a robot dog ........ 46
Csaba Kertész

Partially separable fitness function and smart genetic operators for area-based image registration — Part 2: new operators ........ 54
Janne Koljonen

Online machine vision for elementary engineering courses .......... 62
Janne Koljonen, Mats Björkqvist and Jarmo T. Alander

Symbolic analysis: From theory to practice .......... 70
Erkki Laitila

Solving Rubik’s cube with genetic algorithm .......... 74
Timo Mantere

An online evaluation platform for proactive information retrieval task ........ 79
Li Yao and Antti Ajanki
Foreword

Welcome to the 14th Finnish Artificial Intelligence Conference (STeP 2010). Step 2010 continues the long tradition of the Finnish artificial intelligence conferences of which the first was held at the Helsinki University of Technology (TKK) in August 20–23, 1984. It has been organized regularly every two years ever since. Step 2010 will be held in the Aalto University School of Science and Technology in Espoo on August 17–18, 2010.

Invited Talks

The program of this year’s conference has a strong focus on invited speakers. Namely, we have five invited talks, one tutorial, and a special Program consisting of two demo sessions.

The first of the five invited talks is given by Jarmo Alander from University of Vaasa, whose talk “Computational intelligence in Vaasa”, discusses about the research of automation engineering, evolutionary algorithms, and beyond. Their applications range from various optimization problems in engineering to medical and economic applications.

Tapio Salakoski from University of Turku will talk about “Machine learning and language technology for the bio-health domain”. The talk discusses about general purpose machine learning algorithms, natural language processing methods for the bio-medical domain, and methods for automated syntactic and semantic parsing of Finnish, that have recently been developed by the researchers in University of Turku.

The talk “From an artificial intelligence to multiple subjective intelligences: Approaching the second level of subjectivity”, is presented by Timo Honkela from Aalto University. The talk discusses the challenges in sharing subjective meanings of symbols between individual agents in distributed artificial intelligences.

Ville Kyrki from Lappeenranta University of Technology presents a talk “Machine learning for abstraction in robotic systems”. In his talk, he speaks about the use of machine learning approaches in connecting abstract concepts of inference to sensor readings, connecting signals with symbols.

Harri Valpola from Aalto University and ZenRobotics Ltd. speaks about a new generation of neural networks in his talk “From neural networks to artificial intelligence”. The new generation, while being deeply rooted in neural networks tradition of parallel distributed processing, analog representations and focus on learning from real-world data, relies on emergent dynamic phenomena which correspond to symbols, role binding and manipulation of structured representations.

In addition, Jolanta Mizera-Pietraszko from Wroclaw University of Technology, Poland, presents a tutorial on “Multilingual document mining for unstructured information”. The tutorial takes a top-down approach to understanding the query profiles when mining the multilingual resources for unstructured information.

Finally, Pentti Haikonen, University of Illinois at Springfield, hosts a special program about “New experimental approaches to computation and robot intelligence”. The program consists of two demo sessions of which the first is about “Quasi-quantum computer; the fast computation of factors”, and the second about “A robot with non-digital associative neural processor”.

Contributed Papers

In addition to the abstracts of the invited talks and of the tutorial, the conference proceedings consist of contributed papers. This year, the program committee decided to tighten the review process and to become more selective in the submissions. The conference received altogether 16 submissions. Based on the feedback from the reviewers, 10 of the submissions were accepted for oral presentation at STeP 2010 conference and for publication in the conference
proceedings.

In their paper “GArphics — Applying genetic algorithm for generating graphics”, Anssi Jäntti and Jarmo T. Alander from University of Vaasa introduce GArphics, an interactive computer program designed for image generation. The program is based on mathematical functions that are modified using a genetic algorithm.

The paper entitled “Partially separable fitness function and smart genetic operators for area-based image registration — Part 2: new operators” written by Janne Koljonen from University of Vaasa introduces new genetic operators. The proposed operators, together with genetic algorithms, can be applied for the task of image registration.

The paper “Online machine vision for elementary engineering courses” by Janne Koljonen, Mats Björkqvist and Jarmo T. Alander from University of Vaasa, describes a framework applied in the University of Vaasa for teaching engineering students to design and use field-programmable gate arrays, especially using the VHDL description language.

Timo Mantere from University of Vaasa describes in his paper “Solving Rubik’s cube with genetic algorithm” the problem of solving Rubic’s cube and an genetic algorithm based approach for finding the sequence of moves needed in order to solve the puzzle. Several fitness functions depending on different properties of the state of the cube are tested together with the genetic algorithm.

The paper “Experiments with domain adaptation methods for statistical MT: From European parliament proceedings to Finnish newspaper text” is written by Marcus Dobrinkat and Jaakko Väyrynen from Aalto University. Four different statistical machine translation domain adaptation methods are evaluated using a Finnish-English news corpus. The problem addressed is that due to the limited availability of annotated data, one must often make use of out-of-domain data when building statistical translation models for new domains. Since the distributions for the training and test data are different, the learned translation model must be adapted in order to perform well on the test domain.

Luis Gabriel De Alba Rivera, Alexander Ilin, and Tapani Raiko from Aalto University wrote the paper “Comparison of variational Bayes and Gibbs sampling in reconstruction of missing values with probabilistic principal component analysis” concerning the recovery of missing values from data. MovieLens data and an artificially generated data set are used in an empirical study in which different recovering approaches are compared for the value reconstruction.

The paper by Li Yao and Antti Ajanki from Aalto University, entitled “An online evaluation platform for proactive information retrieval task”, introduces software for aiding the research and development of personalized IR systems. The software automates the gathering of implicit user feedback from eye movements, and the subsequent evaluation of machine learning models trained from such data.

Zaheer Ahmed, Aamir Shahzad and Craig A. Lindley from Blekinge institute of technology, Sweden, have written the paper “Gaze contingent adaptive interface for tele-operation” which introduces a gaze contingent dynamic interface for the control of a mobile robot. Commands to the robot are given by dwelling eye gaze in a specific command region for a specified time. The concept is evaluated in two experiments where the completion time is measured and it is shown that the completion speed is increased with the proposed interface.

The paper “A synchronized system concept and a reference implementation for a robot dog” of Csaba Kertész from Vincit Oy, describes a system concept for robot control. The concept takes advantage of the idea that the computationally heavy data processing can be performed in a remote computer having larger amount of computational resources than the robot hardware contains on board. As a reference implementation, the author presents a control system for the AIBO robot dog.

Erkki Laitila from SwMaster Oy has written the paper “Symbolic analysis: From theory to practice”. In his paper, he presents observation-reasoning-technology (ORT), a systematic methodology for localizing software problems.
Acknowledgements

We are grateful to the invited speakers, who accepted invitations to come and present their work in the conference and to all the active researchers who have submitted their contributions. Moreover, we thank Jolanta Mizera-Pietraszko for presenting her tutorial in STeP 2010 conference. Further, we thank the referees for their prompt reviews of the submitted papers and the program committee for taking care of paper selection.

Tapio Pahikkala (STeP 2010 program committee chair)
Organizing and program committees

- Tapio Pahikkala (program committee chair)
- Jukka Kortela (organizing committee chair)
- Antti Airola
- Pentti Haikonen
- Tapani Raiko
- Jaakko Väyrynen
Computational intelligence in Vaasa

Jarmo T. Alander
University of Vaasa
P.O. Box 700, 65101 Vaasa, Finland

Abstract

Automation engineering education and research at University of Vaasa is computationally oriented. Computational intelligence, especially evolutionary algorithms and their applications, are worked on. The applications range from various optimisation problems in engineering to medical and economic applications. Judged by the number of published papers the automation group is the leading one in evolutionary computing in the Nordic countries [1]. Perhaps the most artificial intelligence flavour has work on evolutionary algorithms related to Sudoku puzzles [2, 3]. In evolutionary software testing the group is among the global top ten most active groups [4]. Medical applications are related to visual and near-infrared imaging [5]. In the medical applications evolutionary optimisation is usually used in spectral wavelength selection (optimal minimum set) of model parameter search/optimisation. University of Vaasa has had own education in engineering since 2004. This year saw the graduation of our first doctoral student, when Janne Koljonen defended his doctoral thesis on machine vision. That work also contained some optimisation done by genetic algorithms [6]. Currently there is one doctoral thesis work going on on medical skin imaging applying also GAs in model parameter search [7].

The future activities is planned to include indocyanine green imaging development and optimisation in surgical applications [8] and online video image processing with FPGAs. The more traditional future engineering applications are expected to include optimisation related to energy production and use. [9].

CV: Jarmo Alander is professor of production automation at University of Vaasa, Department of Electrical and Energy Engineering.

References


From an artificial intelligence to multiple subjective intelligences:  
Approaching the second level of subjectivity

Timo Honkela  
Aalto University School of Science and Technology  
Department of Information and Computer Science  
Adaptive Informatics Research Centre  
Computational Cognitive Systems research group  
timo.honkela@tkk.fi

Abstract

In artificial intelligence and cognitive science research, it has been commonplace to concentrate on understanding and modeling the intelligence of a single agent as a “representative” of all agents as if they were all the same or at least similar enough. On the other hand, the researchers on distributed artificial intelligence have already for a long time paid attention to the social level of knowledge. A single agent may have knowledge of a particular part of some domain or it may be able to solve only particular kinds of problems. These agents then collaborate to solve more complex problems that exceed their individual skills or knowledge. We can call this distribution of skills and knowledge as the first level of subjectivity. However, in the classical AI research this consideration of subjectivity can be deemed to be superficial in a particular manner. Namely, it is usually assumed that the epistemological framework in which the knowledge is represented is shared by the individuals. In other words, the symbolic items in the knowledge representations are assumed to have fixed meanings, shared by all the agents. For instance, if one agent knows that mother(mary, john) and father(john, peter) and another agent knows that grandmother(X, Y) :- mother(X, Z) and father(Z, Y) these agents can together conclude that grandmother(mary, peter). It is notable that family relationships are an example of unusually clear relationships and the underlying concepts are crisp. The research on fuzzy sets has, however, already shown that many real world concepts that underlie their linguistic descriptions have imprecise boundaries. There are multiple methodological alternatives to represent this impreciseness apart from fuzzy sets but the underlying issue is that meanings should be considered as distributions rather than points or nodes in some discrete networks of connected items. The relationships are important but they are not sufficient alone in defining meaning. Let us assume, just for the sake of the argument, that the human mind stores a huge number of membership function value distributions related to a large number of concepts. The fuzzy set theory has some times been criticized for the question how does one define the membership functions in a proper manner as there are so many potential ways of doing it. But, actually this seems to be exactly the empirical situation in each human mind! Namely, each individual has a subjective way to assess the impreciseness of each concept. What is considered red, long, beautiful, democratic or lovely by one individual may not be same for another. The exact interpretation of, for instance, the concept of house, mother or computation depends on the subject and these interpretations are not shared in a manner that could be defined in a simple, logical way. Therefore, one cannot assume that some concept could be represented as a symbol that then somehow miraculously represents its meaning. One may assume that one uses some prototypical meaning of the symbols but that does not solve the underlying problem and does not facilitate the proper contextual use of these symbols in intelligent systems. The meaning of a symbol needs to be grounded as has been pointed out by some researchers already for some time (Harnad 1990, Cangelosi et al. 2000). But this grounding has to be done for each individual separately (and this is typically neglected).
This phenomenon of subjective meaning functions we call the second level of subjectivity. Methodologically, when we represent the meaning as distributions in high-dimensional spaces, grounded at perceptional, action-based and cultural level, the interpretations of symbols can be mathematically and computationally compared (Raitio et al. 2004). Moreover, steps towards the direction of a theory of representation and communication that approves the empirical fact of the second level of subjectivity can be taken (Honkela et al. 2008). In essence, this approach is based on the idea that the agents adapt themselves towards shared symbolic and linguistic representations through imitation in shared contexts that facilitate (partial) agreement of the use of the symbols (Lindh-Knuutila et al. 2006, 2009; see also Steels and Kaplan 2000). In general, considering the second level of subjectivity has many practical implications for the future AI and information systems research and development including, for instance, the possibility of building inter-operable systems that are able to adapt their understanding in changing contexts and communicative situations. Methodologically, sophisticated mathematical and computational methods are needed that match with the complexity of the underlying dynamic phenomena (consider, e.g., Ritter and Kohonen 1989, von Foerster 2002 and Hyötyniemi 2010, including philosophical discussions).

**References**


Abstract

Since the first autonomous robot, Shakey, was built in late 60’s in Stanford, artificial intelligence and robotics have had a long and stormy relationship. Many results of artificial intelligence studies, such as planning algorithms, are essential tools in robotics. However, traditional artificial intelligence approaches have challenges now that robotic systems are appearing more and more in less structured and uncertain environments. The main challenge is to cope with the variability and uncertainty of the world.

The presentation will discuss the use of machine learning approaches in connecting abstract concepts of inference to sensor readings, connecting signals with symbols. Examples of recent research are presented, including topics such as understanding the structure of human motion and abstraction of physical manipulation.

Ville Kyrki is a professor in Computer Science and head of Machine Vision and Pattern Recognition laboratory at the Lappeenranta University of Technology. His research interests lie mainly in intelligent robotic systems and computer vision.
Abstract

Machine learning algorithms enable the automatic construction of methods that would be difficult or laborious to program by human experts. Despite the recent success, there are enormous challenges in deploying these techniques into applications due to the excessive requirements of computational and memory resources. Novel algorithms are needed for training the learning methods and for evaluating the validity of the results. We have developed computationally efficient algorithms for ranking, preference learning, performance evaluation, and feature selection. Our algorithms have found their use in practical applications in bioinformatics and natural language processing.

BioNLP is a research field where natural language processing methods are developed for the bio-medical domain. We have developed a system for automated extraction of complex, recursively nested molecular biology events from scientific publications. The system makes use of modern machine learning algorithms to cope with the large amount of training data and the high-dimensional feature space. We have applied the system to 20 million sentences of text, resulting in 19 million events forming a massive event network that captures much of biomolecular knowledge.

We also study automated syntactic and semantic parsing of Finnish. We study the applicability of efficient, linear-time parsing algorithms fully reliant on machine learning techniques. In close collaboration with public and private partners along the health care value chain, we develop language technology for enhanced understanding and communication of health information such as in electronic patient records.

Tapio Salakoski is a Professor of Computer Science and Head of the Department of Information Technology at University of Turku, and Vice Director of Turku Centre for Computer Science. His research interests are in machine learning and natural language processing for bio-medical and health informatics.
Abstract

Symbolic artificial intelligence and neural networks research have both aimed at intelligent systems but their starting points have been very different. Symbolic AI has focused on high-level cognition, symbol manipulation, reasoning and planning, while neural networks research has studied perception and learning. In this talk I will discuss a new generation of neural networks which are approaching higher levels of cognition. They are deeply rooted in neural networks tradition: parallel distributed processing, analog representations and focus on learning from real-world data. However, unlike most current neural networks, the new generation of networks relies on emergent dynamic phenomena which correspond to symbols, role binding and manipulation of structured representations. In other words, we are again tackling the same problems as symbolic AI but this time we are firmly grounded in real-world data and learning.

Academy research fellow Harri Valpola, PhD, is the leader of computational neuroscience group in Aalto University School of Science and Technology. The group studies the information processing principles of the brain and applies them in robotics. Dr. Valpola is also the chief scientist at ZenRobotics Ltd. which applies the research in intelligent, learning robots.
Multilingual document mining for unstructured information

Jolanta Mizera-Pietraszko
Wroclaw University of Technology, Poland

Abstract

Exploration of documents in multiple languages stimulates a knowledge-specific demand for recognition of linguistic patterns and associations between particular data entities. Pragmatically, these patterns set a trend in the most likely grammar usage, contextual word sensing as well as their frequency of occurrences for a particular language group.

Mining data warehouses for information in federated, heterogeneous environment, or more explicitly, under a scenario in which the unstructured, or semi-structured documents are in many languages, generates a selection of the linguistic patterns with the highest co-occurrence in all the multi-texts simultaneously. Additionally, a comprehensive systematization of mutually consistent morpho-syntactic structures is found a backbone of the trans-lingual systems architecture. Such an approach predisposes to creating a user profile that can optimize the system efficiency by increasing the number of the relevant responses, which are then indexed by matching the key words to these linguistic patterns co-occurring in the documents being mined. This tutorial takes a top-down approach to first understand the query profiles when mining the multilingual resources for unstructured information and is a subject to discussion of the impact of the translation accuracy on the trans-lingual systems’ responsiveness.

Accordingly, this tutorial addresses the issues of trans-lingual engineering objectives which focus on submitting a query in a language other than the system responses and the algorithms of a system ranking score. It will help the research community, both working in the field of trans-lingual engineering as well as in natural language processing, to identify the problem of so called noisy data.

Based on a holistic English language framework the tutorial presents a methodology on how to investigate language pair phenomena in the context of the multilingual document mining, finding it a probable distinctive factor which can also facilitate the translation process and ultimately, the cross-language retrieval.

Furthermore, based on some co-occurrence statistics, the tutorial considers contextual information sensing in the process of a bi-text creation. For building bi-texts the tutorial takes the audience through the most influential challenges in text analysis like e.g. preprocessing a multilingual document collection and the metrics commonly utilized in translilingual data mining. Some evaluation methodology is provided for probabilistic automated translation models.

In the last part, the tutorial proposes an open discussion on the recent advances in the field of trans-lingual engineering and the implementation trends in the real world as opposed to those in the public sector.

Biographical Sketch of the Presenter

Jolanta Mizera-Pietraszko is a Ph.D. Fellow at Institute of Informatics, Faculty of Computer Science and Management, Wroclaw University of Technology, Poland. Her research interests are mainly focused on multilingual search engines, parallel languages, bi-text processing, bilingual question answering systems and multilingual digital libraries.
She invented an innovative language and system independent asymmetric translation technology entitled An Approach to Analysis of Machine Translation Precision by Using Language Pair Phenomena, Invention number P387576 registered on 23.03.2009 by the Patent Office of the Republic of Poland. She in an FP7 Expert for the European Commission in Brussels, an Expert in R&D projects for the Ministry of Science, an Evaluator of English Course books for the Ministry of Education. She gave a tutorial on Translation Component as an Impact Factor on the Retrieval Results at Ionian University, Corfu, Greece, an Association of Computing Machinery Board Member for Computing Reviews, US. She reviews up-to-date software release and books for the British Computer Society, London, UK. She is an IEEE Technical Committee on Digital Libraries Fellow. She has been invited to serve on International Program Committees of the conferences in the UK, Czech Republic, India and Poland. Her projects have achieved recognition from the university, the European Union, foreign scientific institutions and the Polish Ministry of Science.
Gaze Contingent Adaptive Interface for Tele-operation

Zaheer Ahmed⋆†
⋆Blekinge Institute of Technology
SE-371 79 Karlskrona Sweden
virtualtalks@gmail.com

Aamir Shahzad†
†Blekinge Institute of Technology
SE-371 79 Karlskrona Sweden
phulsphy@gmail.com

Craig A. Lindley‡
‡Blekinge Institute of Technology
SE-371 79 Karlskrona Sweden
craig.lindley@bth.se

Abstract

Using eyes as an input modality for different control environments is a great area of interest for enhancing the bandwidth of human machine interaction and provide interaction functions when the use of hands is not possible. Interface design requirements in such implementations are quite different from conventional application areas. Both command-execution and feedback-observation tasks are performed by human eyes simultaneously. In order to control the motion of a mobile robot by operator gaze interaction, gaze contingent regions in the operator interface are used to execute robot movement commands, with different screen areas controlling specific directions. Dwell time is one of the most established techniques to perform an eye-click analogous to a mouse click. But repeated dwell time while switching between gaze-contingent regions and feedback-regions decreases the performance of the application. We have developed a dynamic gaze-contingent interface in which we merge gaze-contingent regions with feedback-regions dynamically. This technique has two advantages: Firstly it improves the overall performance of the system by eliminating repeated dwell time. Secondly it reduces fatigue of the operator by providing a bigger area to fixate in. The operator can monitor feedback with more ease while sending commands at the same time.

1 Introduction

In Human Computer Interaction (HCI) it is critical to study both the context of task performance and the role of attention, including visual focus of attention. A user’s gaze is a strong indicator of intention or attention (Zhai, 2003). Moreover, when functioning well, the eyes are the primary source of information about the context and details in the environment as a basis for action (Kumar et al., 2008). Hence using the eyes as an input modality compared to more conventional input modalities (key-board, mouse, joy-stick etc.) has been an area of great interest in the field of HCI (Levine, 1981; Bolt, 1982; Ware and Mikaelian, 1987; Robert J.K., 1990). Factors that give inspiration and motivation for this include (Zhai et al., 1999):

1. In some of situations both of a user’s hands are constantly engaged with other tasks, and disabled users may not be able to use their limbs.
2. Eye movement is faster than the other parts of the body. The process of acquiring and activating a target point on an interface using a cursor involves two steps: visually focusing the target first and then performing the actuation of the cursor. This means that if we can track the eye gaze successfully and use it accurately and efficiently, no other input source can be as fast as eye.
3. Key-board and pointing devices may be a cause of exhaustion and potential damage. This is another factor of concern in the field of HCI. Eye gaze as an input modality may be a nice solution to these problems.

As eye-tracking systems become more accessible, there are an increasing number of demonstrations of the use of eye-tracking to control the motion of controllable physical agents like robots (Bolt, 1982). Eye-tracking can provide an accurate gaze point on an object on a computer screen that the user is looking at (e.g. Tobii T60). In robot control systems, eye gaze direction can be used as an input source, similar to conventional input modalities: mouse, key-board, joy-stick etc. As such, gaze-based interaction has been used extensively to provide interaction capability for computer users who lack adequate use of their hands and arms, e.g. for word processing, writing email, and using the web (SmartboxAssistiveTechnology, 2010). Gaze contingent interfaces for robots can provide further assistance to those with disabilities, in the form of systems for manipulation (e.g. by directing robot arms) or for exploration (e.g. controlling the movement of mobile
Eye-tracking is also being explored as a method for enhancing human-robot communication, e.g. to allow humanoid robots to react to human eye movements during conversational interactions (Atienza and Zelinsky, 2002). As an example of gaze contingent interfaces for the disabled, (Lin et al., 2006) describes an interface developed to control a powered wheel-chair. This interface is operated by human eye gaze. The interface is divided into 9 regions. Of those 9 regions, 4 are gaze contingent or command regions and remaining 5 regions are idle regions. Stop command is sent to the wheel-chair when user’s gaze falls into these regions. Gaze contingent regions in an eye gaze controlled interfaces are those regions that are used to trigger specific commands when the eye gaze falls within them. In (Lin et al., 2006), screen regions are used to initiate wheel-chair motion commands. Similar work is presented in (Barea et al., 2002), where electrooculography (detection of electrical impulses from muscles that control eye gaze direction) is used to send driving commands to a wheel-chair from the interface. In both of these works, the operator is sitting on the wheel-chair. No feedback is provided to the operator through the interface, i.e. the interface is used only for one-way communication.

An experimental eye-tracking algorithm has also been used to control a robotic arm (Yoo et al., 2002). For this experiment, the interface is divided into 2 regions: a command region and the feedback region. Feedback is provided in the form of images taken by a camera installed in the robot location. Similar work has been presented for control of a robotic toy (Bhuiyan et al., 2004). In this work a different eye-tracking technique is used: in order to find out the gaze direction, location of the eye ball in the eye socket is tracked. In this technique the only purpose of the interface is to present feedback to the operator. In the more general field of robotics (i.e. beyond concerns with human disability) most research addresses controllable agents rather than fully autonomous agents (Olsen and Goodrich, 2003). In most cases these controllable agents are required to be controlled from a remote location, an approach called tele-operation (Latif et al., 2008). TeleGaze (Latif et al., 2008) and Gaze-controlled Driving (Tall et al., 2009) are recent projects in the area of robotics and control systems using eye gaze contingent interfaces for more natural human robot interaction. In TeleGaze, the interface is divided into 9 gaze contingent regions for different commands. These gaze contingent regions take 1/3 second dwell time to activate and issue the associated command to a WIFI enabled modified wheel-chair robot. Dwell time enables eye gaze to act as mouse click. When the operator fixates in a gaze contingent region for a predetermined interval of time (e.g. 1/3 sec), it is considered as an activation similar to a mouse click and a command is issued to the system to be controlled.

Tele-operation is characterized by controlling some system from a remote location (Latif et al., 2008). It involves two separate actions: one is to monitor the current state of the system to be controlled and the second is to send the appropriate commands according to the current state of the system. Monitoring is performed by the human eye and the hands are responsible for command execution through conventional input modalities like a key-board, mouse or joy-stick. If eye gaze alone is used as an input modality for tele-operation, it is obvious that both monitoring and command execution must be performed by eye gaze. The eye can fixate (i.e. focus at a point on the screen to keep a cursor stationary at a gaze contingent region for a certain interval of time) (Lankford, 2000) only on one object at a time; i.e. when the operator tries to focus on feedback (Monitoring) she loses focus on command execution interface, and vice versa (this applies in all of (Lin et al., 2006), (Barea et al., 2002), (Latif et al., 2008) and (Lankford, 2000). The resulting constant switching between two focus areas decreases the performance of the system considerably. The main reason for this performance deterioration is dwell time. Dwell time reinitializes to zero if the operator loses focus within a gaze contingent region in the interface. So switching between feedback-monitoring and command-execution takes repeated dwell time to send commands, resulting in performance deterioration.

An alternative design, described below, is that if an operator fixates in a gaze sensitive region and after the dwell time whole feedback region becomes the command execution region as well, then the overheads of switching between command execution and feedback regions is eliminated. This is explained in detail in the next section.

2 Interface Design

We seek an answer to the following research question in this paper: How to avoid multiple dwell time and improve performance of the overall activity of tele-operation? In order to answer this question we decided to analyze gaze behavior of the participants on a similar kind of interface proposed in related
work studies (Latif et al., 2008), in tasks controlling a wheeled mobile robot with visual feedback provided by a real time video link from a camera mounted on the robot platform. Gaze-directed control areas of the screen are mapped over the user’s view of the video transmission from the robot. The robot can be driven in forward and backward directions, or turned by control of its differential drive system. Figure-1 is a scatter plot of a user’s gaze fixations in different parts of the screen while interacting with an interface using fixed areas for dwell-activated command initiation. The participant users have used all the gaze contingent regions according to their needs. An interesting observation is that participants have tended to prefer the upper half areas of the RIGHT and LEFT regions. These regions are represented by ⬤ and ⬦ in the plot, respectively. We used this observation in our alternative interface design and shortened these regions by 70 pixels from below, using this region instead for the STOP command in our alternative interface (Figure-2). Gray portions in the plot represent all those fixations when the participant focuses in the feedback region or some other regions in the monitor screen than gaze contingent command regions. In the new design, the STOP region dynamically adjusts its position in the interface. Initially it rests on top of the BACKWARD region. If a participant fixates in the BACKWARD region it changes its position and sticks to the FORWARD region at the top of interface. Whenever participant user switches between FORWARD and BACKWARD command regions, STOP region moves in between, since it is logical to stop before moving forward or backward. Our interface consists of 10 gaze contingent regions in total: 2 regions for STOP, 4 regions for FORWARD, BACKWARD, RIGHT and LEFT. The remaining 4 are dynamically formed by expansion of FORWARD, BACKWARD, RIGHT and LEFT. All other regions except STOP expand over the whole feedback region after the dwell time when fixated by the user. This provides more ease to the participant user since it is easy to fixate in comparatively larger regions. It also eliminates repeated dwell time for multiple commands. When one task is completed, the user fixates in some other gaze contingent region to perform a task associated to that region. As a result, the previous gaze contingent region shrinks back to its default location and the new region as selected by the user merges with the feedback region, and so on. This is a more appropriate technique for tasks that need multiple commands (e.g. turning to a suitable angle towards the right in multiple increments of 15°).
3 Method

Our empirical study is based on two experiments. Both the experiments are task-oriented. Human robot interaction applications are very diverse. Hence, there are no standard metrics for evaluation of newly developed applications (Latif et al., 2008). However, there are some common metrics in any application domain that are most likely to be used to evaluate the application developed in that particular domain (TobiiTechnologiesAB, 2010). Time needed to complete a task is a common measure (Tsui et al., 2009).

Experiment 1 is a rather small experiment. The task in this experiment is to turn the robot at an angle of $90^\circ$. The reason for this separate experiment is that the robot we are using can turn only with a limited angle of $15^\circ$ per turn command. In order to get larger turn angles, multiple turn commands are needed; e.g. with a command turn angle of $15^\circ$ the robot needs 6 turn commands to turn a total angle of $90^\circ$. Forward and backward movements are far simpler than turning, since forward and backward motion is continuous. That is why we decided to examine the turning motion in a separate experiment. In experiment 2, a track is designed for the robot to travel on. An experiment participant interacts with the robot using gaze contingent interfaces and navigates through this track. Participants perform both of the experiments multiple times. Data is collected by noting the task completion time for each experiment trial.

3.1 Experiment Design

In our empirical study the independent variable is screen adaptiveness, which has two possible values: static and adaptive. These interface variants are used to control the navigation of a mobile robot. In both experiments participants complete a navigation and movement task. The outcome/dependent measure of the experiments is the task completion time. The upper half of Figure-3 shows the experiment 1 setup. A card of gray color is placed in front of the robot. Another card of black color is placed on the left side of the robot at an angle of $90^\circ$ from the gray card. When a participant fixates in the gaze contingent region specified for LEFT, the robot starts turning to the left. When the turning robot completes an angle of $90^\circ$ it comes in front of the black card. Now the participant can see the black card in the feedback region of the interface and should use gaze in the stop region of the interface to stop further movement. The participant performs this activity with both static and adaptive versions of the interfaces. In experiment 2, participants are given a task to send commands to the robot and monitor visual feedback, to drive the robot around a specified track to return to the starting location. The lower half of Figure-3 presents the layout of the track used in experiment 2.

We observed that the majority of the participants were not familiar with the concept and technology of eye tracking. This was a potential validity threat and could have effected the outcome due to varying capabilities and understanding of the participants. To bring them at an equal level of understanding we arranged training sessions for the participants. Another validity threat was power supplies to the logic and actuation parts of the mobile robot. Weak batteries may result in slow movement and delayed responses. To avoid this threat all the batteries were replaced with a fresh and fully charged set of batteries before switching to the next participant. XBee transceivers are used for communication between application program and mobile robot, having a range of 120 meters without obstacles. We ensured that the mobile robot remained within this range while performing tasks. The track is designed keeping in mind that it

![Figure 3: Task layout for experiment 1 and 2](image)
should include all possible navigational moves. The total length of the track is 19.25 meters. The start and finish point are same. The whole track is marked with black and gray arrows. The participant follows black arrows to move down the track to the turning area and gray arrows for coming back to the finish point. 15 participants in total take part in these experiments. In experiment 1, 10 trials are recorded for each participant. This results in \( 15 \times 10 = 150 \) trials of data. Experiment 2 is a bit lengthier in terms of time. Hence a block of 3 trials is recorded for each participant, giving \( 15 \times 3 = 45 \) trials in total.

### 3.2 Apparatus and Conditions

An Intel® Core 2™ Extreme Q6850 @3.00GHz was programmed using Microsoft® Visual C++™ to create the gaze-directed monitoring and control interface. A Tobii® Eye Tracker T60™ was used to get participant’s gaze behaviour and this was integrated with the interaction system via an API in order to use gaze data on the monitor screen. GrabBee® video grabber was used to capture the video stream from the wireless camera mounted on the head of the mobile robot. Intel’s® image processing library OpenCV® was used to grab the video stream and then superimpose interface components on it. The Arduino® open-source electronics prototyping platform was used to configure and map robot actuation via different commands. XBee® serial communication transceivers were used to communicate commands between the user application software and the mobile robot. The interfaces developed in the project were flexible to work on any resolution but all experiments were conducted with 800x600 screen resolution.

### 3.3 Procedure

When participants arrived they were briefed about the experiments so that they could have an idea about the whole activity to be performed. Participants were seated 60-65 cm viewing distance from the eye-tracking device as directed in the Tobii® T60 user’s manual. A calibration process was performed for each participant to get accurate gaze fixation points on the eye-tracking screen. Then eye tracking was started. Each participant was asked whether the cursor is moving to the desired location of the screen with her gaze fixation or not. When the participant was satisfied with the result, the robot camera view with the interface superimposed upon it was presented to the participant. This concludes the experiment setup and now the participant can fixate in the gaze contingent regions to send commands to the robot. Each participant was allowed 3-4 training sessions to gain familiarity with the environment. Outcomes for actual trials were then recorded according to the pre-planned requirements.

### 3.4 Participants

Fifteen participants took part in this experiment. All were from Blekinge Institute of Technology (BTH). Out of fifteen, 10 were male and 5 female. Two faculty members also took part in this experiment. The age range of the participants was 20-51 years. In order to ensure that all the participants had the same level of understanding they were briefed in the same way about the experiment. They were also given a training session before data collection was started.

### 4 Results

Our claim is that mean task completion time for the dynamic interface design is less than for the static interface design. For this claim we have following null hypothesis (Ho) and alternate hypothesis (Ha).

\[
\text{Ho: Mean task completion time is same for both dynamic and static interfaces.}
\]

\[
\text{Ha: The dynamic interface takes less time to complete a task compared to the static interface.}
\]

In both the experiments the sample sizes are greater than 30, which is large enough to substitute sample variance with population variance. So we use a \( z \)-test (\( \alpha = 0.05 \)) for statistical analysis. Table-1 and Table-2 show the results. In both experiments, \( z \)-stat is less than \( z \)-critical. So we can say that the difference between the populations is statistically significant. Hence the null hypothesis can be rejected in favour of the claim that dynamic interface takes less time to complete a task as compared to the static interface.

<table>
<thead>
<tr>
<th>Description</th>
<th>Dynamic</th>
<th>Static</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>2.966666667</td>
<td>3.976666667</td>
</tr>
<tr>
<td>Known Variance</td>
<td>0.061163</td>
<td>0.567707</td>
</tr>
<tr>
<td>Observations</td>
<td>150</td>
<td>150</td>
</tr>
<tr>
<td>Hypothesized Mean Difference</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( z )</td>
<td>-15.59863388</td>
<td>0.000000000</td>
</tr>
<tr>
<td>( P(Z &lt; z) ) one-tail</td>
<td>0.000000000</td>
<td>0.000000000</td>
</tr>
<tr>
<td>( z ) Critical one-tail</td>
<td>1.644853627</td>
<td>1.959963985</td>
</tr>
<tr>
<td>( z ) Critical two-tail</td>
<td>1.959963985</td>
<td>1.959963985</td>
</tr>
</tbody>
</table>

Table-1: \( z \)-test results for experiment-1

In both experiments the sample sizes are greater than 30, which is large enough to substitute sample variance with population variance. So we use a \( z \)-test (\( \alpha = 0.05 \)) for statistical analysis. Table-1 and Table-2 show the results. In both experiments, \( z \)-stat is less than \( z \)-critical. So we can say that the difference between the populations is statistically significant. Hence the null hypothesis can be rejected in favour of the claim that dynamic interface takes less time to complete a task as compared to the static interface.
Table-2: z-test results for experiment-2

<table>
<thead>
<tr>
<th>Description</th>
<th>Dynamic</th>
<th>Static</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>115.2</td>
<td>153.688889</td>
</tr>
<tr>
<td>Known Variance</td>
<td>2810.618182</td>
<td>3026.491919</td>
</tr>
<tr>
<td>Observations</td>
<td>45</td>
<td>45</td>
</tr>
<tr>
<td>Hypothesized Mean Difference</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>z</td>
<td>-3.379424033</td>
<td></td>
</tr>
<tr>
<td>P(Z&lt;z) one-tail</td>
<td>0.000363189</td>
<td></td>
</tr>
<tr>
<td>z Critical one-tail</td>
<td>1.644853627</td>
<td></td>
</tr>
<tr>
<td>P(Z&lt;z) two-tail</td>
<td>0.000726379</td>
<td></td>
</tr>
<tr>
<td>z Critical two-tail</td>
<td>1.959963985</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4: Experiment-1 results.

Figure 5: Experiment-2 results.

5 Conclusion and Discussion

In this paper we have introduced the concept of a gaze contingent dynamic interface to control a mobile robot. The basic purpose was to decrease the repeated dwell time of gaze directed robot control resulting from switching between feedback and gaze contingent regions. A second purpose of the study was to facilitate the operator in such a way that she feels more comfortable with feedback monitoring tasks while sending navigation commands to the robot at the same time. Results of our pilot study show that the performance of our proposed dynamic interface is significantly better than the static interface. Twelve out of fifteen participants reported that they felt more comfortable with using the dynamic interface. The rest of the participants voted in favour of the static interface. An interesting observation in this regard is that the participants who voted in favour of the static interface performed almost same with both interfaces i.e., their task completion time was almost same for both interfaces. All those participants who can drive or play computer games completed experiment task in less time while avoiding collisions, compared to the rest of the participants.

In almost all trials each participant performed quite well in every subsequent trail i.e., she took less task completion time.

Another interesting observation can be made regarding training of the participants. In initial trails we handled each participant alone at the experiment site and she learned from her own experience. But later in the experiments we worked with groups of three or four participants on the experiment site at the same time. In this scenario when each participant was performing the experiment, the remaining were watching her activities. All such participants who watched others, performed exceptionally well on their turn and produced less task completion times. This phenomenon can be seen in graphical results of experiment 2 (Figure-5). It is evident from the graph that task completion times for the initial 12 trials are very high and then we can see more consistent results for rest of the trials. This later region after 12 trials is the region where participants were present in groups.

In the current scenario each gaze contingent region sends a single command to the robot. There is very little variation available in different activities. For example we can move left or right with a fixed angle or forward or backwards with a single speed. The dynamic interface has a quite big area. In a future variant of the interface this area could be used to bring variation in the degree of motion specified in commands. For example, the upper half area of the LEFT gaze contingent region can be used to turn at different angles. or the FORWARD region could be used to move with different speed levels. This may be explored in future work.
References


Comparison of Variational Bayes and Gibbs Sampling in Reconstruction of Missing Values with Probabilistic Principal Component Analysis

Luis Gabriel De Alba Rivera⋆ 
⋆Aalto University School of Science and Technology 
Department of Information and Computer Science 
luis.dealbar@gmail.com

Alexander Ilin† 
†Aalto University School of Science and Technology 
Department of Information and Computer Science 
firstname.lastname@tkk.fi

Tapani Raiko†

Abstract

Lately there has been the interest of categorization and pattern detection in large data sets, including the recovering of the dataset missing values. In this project the objective will be to recover the subset of missing values as accurately as possible from a movie rating data set. Initially the data matrix is preprocessed and its elements are divided in training and test sets. Thereafter the resulting matrices are factorized and reconstructed according to probabilistic principal component analysis (PCA). We compare the quality of reconstructions done with sampling and variational Bayesian (VB) approach. The results of the experiments showed that sampling improved the quality of the recovered missing values over VB-PCA typically after roughly 100 steps of Gibbs sampling.

1 Introduction

Human preferences (the quality tags we put on things) are language terms that can be easily translated into a numerical domain. We could assign low values to odd things and high values to enjoyable things, i.e.; rate things according to our experience. These ratings serve us to easily (and grossly) classify and order our preferences from the ones we like the most to the ones we dislike the most. Of course we are limited: we can not rate what we do not know, however; it may be of our interest to know the possible ratings of these unknowns.

In this project we will be working with large and sparse matrices of movies ratings. The objective will be to recover a subset of the missing values as accurately as possible. Recovering these missing values equal to predicting movies ratings and, therefore; predicting movies preferences for different users. The idea of correctly recovering movies ratings for different users has been a hot topic during the last years motivated by the Netflix prize.

The concept of mining users preferences to predict a preference of a third user is called Collaborative Filtering, it involves large data sets and has been used by stores like Amazon and iTunes.

We can start by considering that the preferences of the users are determined by a number of unobserved factors (that later we will call components). These hidden variables can be, for example, music, screenplay, special effects, etc. These variables weight different and are rated independently, however; they, together, sum up for the final rating, the one we observe. Therefore; if we can factorize the original matrix (the one with the ratings) in a set of sub-matrices that represent these hidden factors, we may have a better chance to find the components and values to recover the missing ratings [1]. One approach to find these matrices is to use SVD (Single Value Decomposition), a matrix factorization method. With SVD the objective is to find matrices $U$ $V$ minimizing the sum-squared distance to the target matrix $R$ [2].

For this project we consider matrix $Y$ to be our only informative input. Matrix $Y$ is, usually, large and disperse, i.e.; with lots of missing values. The observable values are the ratings given to movies (rows) by users (columns). Our objective is to recover the missing values, or a subset of them, with a small error. We can factorize matrix $Y$ such that

$$Y \approx WX + m,$$

(1)

where the bias vector $m$ is added to each column of the matrix $WX$. Matrices $WXm$ will let us recover the missing values, of course, the quality of the recovering depends on the quality of these matrices. Sampling will let us improve the fitness of matrices.
\[ W \times m \] to better recover matrix \( Y \). We can use VB-PCA (Variational Bayes PCA) for the initial decomposition of the input matrix \( Y \). VB-PCA is known to be less prone to over-fitting and more accurate for large-scale data sets with lots of missing values compared to traditional PCA methods [3, 4]. However; VB-PCA is not compulsory for sampling, a random initialization method is also explored in this project.

2 Sampling PCA

Sampling can be seen as the generation of numerical values with the characteristics of a given distribution. Sampling is used when other approaches are not feasible.

For high-dimensional probabilistic models Markov chain Monte Carlo methods are used to go over the integrals with good accuracy. Gibbs sampling is a well known MCMC method [5, 6]. In Gibbs approach we sample one variable, for example \( W \), conditioned to the remaining variables, \( X \) \( m \). In the following step we sample another variable fixing the rest; we repeat this process generating as many samples as necessary.

In our project we have matrix \( Y \) that is a joint distribution of the form \( Y = WX + m + \text{noise} \) to predict the missing values in \( Y \) we need to solve:

\[
P(Y_{MIS} \mid Y_{OBS}) = \int P(Y_{MIS} \mid W, X, m) P(W, X, m \mid Y_{OBS}) dW dX dm.
\]

Solving the integral is complex, therefore; we use Gibbs sampling to approximate its solution. To recover matrices \( W \) \( X \) \( m \) we need to solve \( P(W \mid Y_{OBS}, X, m) \), \( P(X \mid Y_{OBS}, W, m) \) and \( P(m \mid Y_{OBS}, W, X) \) each one following a Gaussian distribution, contrary to \( P(W, X, m \mid Y_{OBS}) \) that follows an unknown and complex distribution. The mean matrices, \( \bar{X} \) \( \bar{W} \) \( \bar{m} \), and covariance matrices, \( \Sigma_x \) \( \Sigma_w \) \( \Sigma_m \), are calculated according to the formulas provided at [4] Appendix D; this is done as follows:

\[
\bar{X}_j = (\bar{W}_j^T \bar{W}_j + vI)^{-1} \bar{W}_j^T (Y_j - \bar{m}_j) \tag{3}
\]

\[
\Sigma_{x,j} = v(\bar{W}_j^T \bar{W}_j + vI)^{-1} \tag{4}
\]

\[
\bar{W}_i = (Y_i - \bar{m}_i)^T \bar{X}_i (\bar{X}_i^T + v \text{diag}(w_i^{-1})) \tag{5}
\]

\[
\Sigma_{w,i} = v(\bar{X}_i \bar{X}_i^T + v \text{diag}(w_i^{-1})) \tag{6}
\]

Indices \( j = 1, \ldots , p \) and \( i = 1, \ldots , m \) go over the rows (people) and columns (movies) of matrix \( Y \), and \( y_{ij} \) is the \( ij \)th element of matrix \( Y \). \( X_j \) is the column \( j \) of matrix \( X \), \( W_i \) is row \( i \) of matrix \( W \), \( m_i \) is element \( i \) of vector \( m \). \( v \) and \( w_m \) are hyper-parameters. \( Y \) is the data matrix where the missing values have been replaced with zeroes. \( O \) is the set of indices \( ij \) for which \( y_{ij} \) is observed. \( O_i \) is the set of indices \( j \) for which \( y_{ij} \) is observed. \( |O_i| \) is the number of elements in \( O_i \). \( I \) is the identity matrix. \( \text{diag} \) is the diagonalizing of the referred values. \( W_i \) is matrix \( W \) in which an \( ith \) row is replaced with zeros if \( y_{ij} \) is missing, \( m_j \) is vector \( m \) in which each \( ith \) element is replaced with zero if \( y_{ij} \) is missing, and \( X_i \) is the matrix \( X \) in which a \( jth \) column is replaced with zeros if \( y_{ij} \) is missing.

Using the mean and covariance matrices we are able to sample \( W' X' \) and \( m' \) using the methods presented in [6]. With the sampled and mean matrices we recover a full matrix \( Y' \), i.e.; including the missing values; more of this is explained in the following subsections.

2.1 Recovering the Missing Values

To recover the matrix \( Y \) we need to multiply matrix \( W \) by \( X \) and add the \( m \) bias vector to each column. Referring to the ideas presented by [1], matrix \( W \) represents the different and weighted factors that conform a movie. On the other hand, matrix \( X \) represents the values assigned to each factor by the different users. The resulting matrix \( Y' \) has, therefore, the ratings given to movies \( m \) by users \( p \). The bias term, \( m \), is used to compensate the differences in results from the recovered matrix \( Y' \) and the original observed values used during the training.

To prove the quality of the ratings in the recovered matrix \( Y' \) it is necessary to have a test set different from the training set. At every step during sampling when the values are recovered we calculate the Root Mean Square Error, RMSE, using the test set as baseline. RMSE is a well known measure to quantify the amount by which a predictor differs from the value being predicted.

The sampling and recovering process is as follows:

1. Start point \( i = 1 \), with matrices \( W^i X^i \) and \( m^i \).
2. Calculate mean matrix \( X \) and covariance matrix \( \Sigma_x \) using \( W \) by Eqs. (3)-(4).
3. Recover $Y'$ with $W^i$ and $\bar{X}$ by Eq. (1).
4. Increase $i$ by one.
5. Sample $X^i$ using from $N(\bar{X}, \Sigma_x)$.
6. Calculate mean matrix $\bar{W}$ and covariance matrix $\Sigma_w$ using $X^i$ by Eqs. (5)–(6).
7. Recover matrix $Y'$ with $\bar{W}$ and $X^i$ by Eq. (1).
8. Sample $W^i$ from $N(\bar{W}, \Sigma_w)$.
9. Calculate bias mean $\bar{m}$ and variance $\tilde{m}$ using $W^i X^i$ by Eqs. (7)–(8).
10. Sample bias $m^i$ from $N(\bar{m}, \tilde{m})$.
11. Loop from step 2.

This can be graphically visualized at Figure 1. At every loop, when calculating the mean matrices $\bar{W}$ and $\bar{X}$ (steps 2 and 6), we use the original matrix $Y$, this leads to an improvement in the recovered values (better representing the original matrix with the observed values) and hence improvement in the future sampled matrices.

Figure 1: Sampling PCA process.

Every time matrix $Y'$ is calculated (steps 3 and 7) the missing values are recovered. At every recovering step the missing values are averaged with the previously recovered ones

$$y^{k+1} = \frac{k y^k + y^{k+1}}{k + 1},$$

where $k$ is the step, $\bar{y}$ is the average of the previous values and $y$ are the new recovered values. Using the average will lead to better results than just using the single-samples alone. The more samples are averaged, the closer the approximation is to the true integral in Equation 2.

3 Tests and Results

The Sampling PCA method was tested with an artificial data set and the MovieLens data set. For the MovieLens test the missing values were also predicted randomly to observe how close a random prediction is from the sampling approach, i.e.; to grossly measure the benefit of using sampling. With the artificial data we will focus on recovering all missing values while with Movielens data only a subset of the missing values.

3.1 Artificial Data

The initial testing was done using artificially generated data. The artificial data consists of generating matrices $W[m, c]$ (normally distributed $N(0, 1)$, random values); $X[c, p]$ (uniformly distributed $[0 \ldots 1]$, random values) and, an additional noise matrix $N[m, p]$ (normally distributed $N(0, \text{var})$ where noise variance (var) is given in the table below). Matrix $Y[m, p]$ is generated as $Y = WX + N$. From matrix $Y$ a given percentage of ratings is selected at random and set to NaN, i.e.; set to be missing values.

Three data sets were generated with the following characteristics:

<table>
<thead>
<tr>
<th>Set</th>
<th>m</th>
<th>p</th>
<th>c</th>
<th>Noise Var</th>
<th>Missing Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>100</td>
<td>125</td>
<td>8</td>
<td>0.05</td>
<td>50%</td>
</tr>
<tr>
<td>B</td>
<td>150</td>
<td>200</td>
<td>15</td>
<td>0.3</td>
<td>70%</td>
</tr>
<tr>
<td>C</td>
<td>300</td>
<td>450</td>
<td>18</td>
<td>0.5</td>
<td>85%</td>
</tr>
</tbody>
</table>

Using the VB-PCA approach, PCA FULL function [4], we recover $W, X$ and $m$ (plus hyper-parameters) from matrix $Y_t$. We do this using 10, 20 and 30 components. With the recovered matrices we run the Sampling PCA algorithm; 500 samples are generated from each input.

We can observe at Table 1, how the noise, size and proportion of missing values of the original matrix $Y$ affect the quality of the recovered missing values. It is also noticeable that when the problem is simple, as...
it is in with data set A, PCA_FULL recovers the matrix with a small error, therefore; no improving can be expected, or achieved, when sampling. On the other hand, with data set C, where the missing values are many and the matrix is noisy and large the recovering achieved from PCA_FULL is just good but it is improved with the Sampling PCA algorithm. An important value affecting the results is the number of components, $c$. Because we do not know the original number of components we try with 10, 20 and 30, and notice that as we get closer to the original number of components our results improve. At Figure 2, are the sampling RMSE error progress through 500 samples compared to the PCA_FULL RMSE error using the best results within each data set.

From the artificial testing we can conclude, first; the number of components used play an important role and, second; as more complex is the problem better results can be expected when using Sampling PCA.

### 3.2 MovieLens Data

The MovieLens [7] data set consist of 100,000 ratings given by 943 users to 1682 movies. Each rating is a triplet, the value of the rating, the user giving the rating, and the movie being rated. The ratings go from 1 to 5, not all movies have been rated nor all users have given rates. Having 100,000 ratings mean that less than 10% of the total possible triplets are available. The data set was divided into Training $Y_t$ and Probing $Y_p$ sets after empty columns/rows were removed, i.e.; users without ratings or movies no rated. The Training set is a matrix of 943x1674 and contains 95,000 ratings. The Probing set is a matrix of the same size but contains, only, 4999 ratings.

The first step consists on recovering matrices $W^\prime$, $X^\prime$ and $m^\prime$ (and hyper-parameters) from matrix $Y_t$ using the VB-PCA implementations PCA_FULL and PCA_DIAG, using 10, 20 and 30 as number of components. The RMSE of the recovered matrix, $Y^\prime$ against $Y_t$ and $Y_p$ can be seen at Table 2. PCA_FULL performed better with the Training matrix while PCA_DIAG was better for the Probing values. For both approaches more components mean worse results against the Training set but better against the Probing one.

With the recovered matrices and hyper-parameters we perform Sampling PCA. Two options are explored, the first option consist in using all the recovered data as starting point for sampling. The second option consists on only using the hyper-parameters; $W^\prime$, $X^\prime$ and $m^\prime$ matrices are initialized with random values.

#### 3.2.1 Sampling From PCA Full/Diag

In this first approach sampling is performed using the recovered matrices and hyper-parameters. For each set of variables 2000 samples are generated, the numeric results can be observed at Table 3. Results show an improvement compared to the $Y^\prime$ vs $Y_p$ RMSE values at Table 2. The use of 20 components seems to return the best results, also, the use of PCA_DIAG shows better results. The best results (shadowed) represent a small improvement, less than
1% against the top result obtained using the VB-PCA approach alone (shadowed at Table 2). However; a small improvement for recovering missing values tasks its an important gain.

At Figure 3, we can observe the RMSE value of each sample through the 2000 samples taken, with different number of components and using PCA\_FULL data as baseline; the values are compared against the RMSE of VB-PCA approach. At Figure 4, a similar plot is observable but in this case using PCA\_DIAG data as baseline. For both Figures, in all sub-plots, we can notice that the sampling algorithm is unstable for the initial samples, the RMSE value jumps around the RMSE recovered from the VB-PCA approach. However; for the last hundreds of samples stabilization is noticeable, showing small differences after each sample.

### 3.2.2 Sampling Using Random Initialization

Another approach to perform Sampling PCA consist in only using the hyper-parameters recovered from PCA\_FULL/DIAG. Matrices $W'X'$ and $m'$ are randomly initialized (uniformly distributed values $[0 \ldots 1]$). This is possible because the algorithms used to recalculate matrices $W'X'$ and $m'$ and their covariances take into account the training matrix $Y_t$. At each iteration of the sampling the matrices $W'X'$ and $m'$ values are updated to better fit $Y_t$.

The initial samples will be highly deviated from the objective value, therefore; they can be eliminated before the real prediction is made. In our test we remove the initial 30 samples. Later, we generate 1000 new samples to make the predictions of the missing values. Again 10, 20 and 30 components are used and the hyper-parameters from PCA\_FULL/DIAG. The Figure 5, shows the discarded samples and how spread they were compared to the final RMSE. The first 10 samples are the most disperse ones, latest samples are more stable in their RMSE value, specially, when the number of components is 20 and 30.

### Table 3: RMSE results in the MovieLens problem after sampling (2000 samples).

<table>
<thead>
<tr>
<th></th>
<th>c=10</th>
<th>c=20</th>
<th>c=30</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA_FULL</td>
<td>0.888123</td>
<td>0.887418</td>
<td>0.887837</td>
</tr>
<tr>
<td>PCA_DIAG</td>
<td>0.884606</td>
<td>0.883733</td>
<td>0.884129</td>
</tr>
</tbody>
</table>

### Table 4: RMSE results after sampling, random initialization.

<table>
<thead>
<tr>
<th></th>
<th>c=10</th>
<th>c=20</th>
<th>c=30</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA_FULL</td>
<td>0.887070</td>
<td>0.887121</td>
<td>0.886675</td>
</tr>
<tr>
<td>PCA_DIAG</td>
<td>0.885025</td>
<td>0.884074</td>
<td>0.885066</td>
</tr>
</tbody>
</table>
4 Conclusions

This project lead to interesting results. The artificial tests let us know that small matrices with small portion of missing values are not easily improved by sampling. For the MovieLens test we observed that sampling improved the quality of the recovered missing values over VB-PCA using the later as an initial step. We also noticed that the random initialization does not affect sampling and the results are good. The best results were obtained using PCA_DIAG and 20 components; the worst results were obtained using PCA_FULL and 10 components. A future improvement could be achieved rounding the recovered values that are outside the range of the expected ones, i.e.; values ≤ 1 to 1 and ≥ 5 to 5. A look at the recovered vector, for the best results, shows 6 values below 1 and 32 above 5.

5 Acknowledgments

Luis De Alba is supported by the Program Al-Ban, the European Union Program of High Level Scholarships for Latin America, scholarship No. E07M402627MX. Alexander Ilin and Tapani Raiko are supported by the Academy of Finland.

References


Dataset: http://www.grouplens.org/node/73

Marcus Dobrinkat and Jaakko J. Väyrynen
Aalto University School of Science and Technology
Department of Information and Computer Science
{marcus.dobrinkat,jaakko.j.vayrynen}@tkk.fi

Abstract

Statistical machine translation methodology is highly dependent on relevant parallel texts for training. However, available large parallel corpora are typically out-of-domain for many interesting translation tasks, such as news translation. We experiment with a very small Finnish–English news corpus using four different domain adaptation methods: language model adaptation, translation model adaptation, automatic post-editing and re-training with combined data. Translation quality is measured with the de-facto standard MT evaluation metric BLEU and we provide statistical significance testing for system comparison. Language model adaptation did not produce significant improvements. All other tested methods outperformed the baseline system.

1 Introduction

Machine translation (MT) adaptation aims to improve translation performance on text in a specific domain that is not present or pronounced in the bilingual training corpus. Domain adaptation is especially important in statistical machine translation (SMT) systems which are trained with empirical data and are closely tied to the training data domain. Text corpora can be very different in many aspects, such as vocabulary, style or grammar. Therefore, the performance of SMT systems is more susceptible to domain differences than traditional rule based systems which do not depend on example translations.

Our task is to improve the quality of a baseline SMT system with different domain adaptation methods. We use a very small in-domain corpus which makes training and evaluation challenging. We provide statistical significance testing to facilitate system comparison.

Our in-domain data consists of a small parallel Finnish–English news domain corpus that we have collected. The translations from Finnish to English were created by volunteers in an online system. The baseline system is trained on the out-of-domain Europarl parallel corpus.

The very small size of the in-domain corpus is highly problematic for parameter optimization and evaluation purposes as separate test and development sets would significantly reduce the size of the training data which would make the model learning very problematic. Our compromise to this was to use cross validation and bootstrap resampling for evaluation and to only try to get reasonable parameter values for the most interesting parameters related to domain adaptation. We believe this is a reasonable approach as we are interested in improving translation quality with different methods rather than optimizing the systems for the best possible performance.

We experiment with four different domain adaptation methods: language model (LM) adaptation, translation model (TM) adaptation, automatic post-editing (APE) and re-training of the model with combined in-domain and out-of-domain data. Language model adaptation does not modify the translation model, but influences the choices made by the translation model, whereas translation model adaptation includes new translations and can be further combined with LM adaptation methods. Automatic post-editing does not modify the baseline translation model, but learns an additional translation model from the output of the baseline system to the reference translations.

In our goal to improve the baseline system translation quality, we evaluate the performance of four different domain adaptation methods, namely language model adaptation, translation model adaptation, automatic post-editing and combination of in- and out-of-domain data for model retraining.

Each tested adaptation method significantly outperformed the baseline system in BLUE scores, except for the LM adaptation in which the improvement was not significant. There was no clear preference of the adaptation methods.
1.1 Statistical Machine Translation

In statistical machine translation, a statistical model governs the mapping from source (s) to target (t) sentence. Although the original ideas of SMT were already introduced in the work of Weaver (1949), the influential work of Brown et al. (1994) renewed the research in the SMT paradigm. The translation problem is represented in a probabilistic framework in which the Bayes formula gives the source-channel approach

\[ P(t|s) \propto P(s|t)P(t) \]  

for machine translation. It splits the conditional probability of \( t \) given \( s \) into two parts: a translation model \((P(s|t))\) and a language model \((P(t))\). The best translation is found by maximizing Equation 1.

A more general approach using a log-linear model, employing a maximum entropy framework has been formulated in Och and Ney (2001). It provides \( M \) feature functions \( h_m(t, s) \) with weights \( \lambda_m \). The translation probability \( P(t|s) \) is then defined as:

\[ P(t|s) \propto \exp \left[ \sum_{m=1}^{M} \lambda_m h_m(t, s) \right] \]  

As before, the best translation is found when \( P(t|s) \) is maximized. The source-channel model in Equation 1 can be modeled as a special case of the log-linear framework in Equation 2 by choosing equal weights, \( \lambda_1 = \lambda_2 \), and feature functions \( h_1(t, s) = \log P(s|t) \) and \( h_2(t, s) = \log P(t) \).

More details about how the model is discriminatively trained, can be found in Och and Ney (2001). One advantage of this more general model is that additional features can easily be included Koehn et al. (2003); Och and Ney (2001).

2 Related Work

Domain adaptation has been performed with a wide range of methods, which can be categorized by their use of in-domain resources (no additional in-domain data, monolingual in-domain data, dedicated in-domain dictionaries or dedicated in-domain parallel corpora) or the way these resources are used (interpolation of out-of-domain with in-domain data or models for language model or translation model adaptation or out-of-domain and in-domain system combinations).

Improving in-domain performance with a dedicated in-domain bilingual corpus is done by Ueffing et al. (2007), who call their approach transductive learning. Using the non-adapted system, they first translate a source language monolingual text corpus, select the good translations and paired them with their source sentences to build a new synthetic in-domain corpus. Re-training the system with this corpus strengthens valuable phrase table content and weakens less useful content, which makes the system gain knowledge from its own output.

Hildebrand et al. (2005) compile an in-domain corpus out of a large general domain bilingual corpus. Their basic assumption is that this large corpus contains different domain sub-corpora, which are obtained by selecting those sentence pairs only, which match the in-domain test set.

Xu et al. (2007) assume an existing bilingual in-domain corpus describing an approach towards a multi-domain machine translation system. The different domain translation models are trained and optimized separately and combined during decoding as different features in a log-linear model. Feature weights are chosen on-line, depending on the domain of the input text.

There are various approaches to LM adaptation, as enumerated by Béchet et al. (2004). He lists linear interpolation of out-of-domain and in-domain language models, and an information retrieval approach where documents matching the required domain are retrieved and trained on-line to create the in-domain language model.

The work of Zhao et al. (2004) combines language model adaptation and information retrieval in the context of machine translation. Using the nonadapted system, they generate a list of translation hypotheses, which are used to create a retrieval query run against large-scale monolingual text corpora. The best result sentences are then used to train a new in-domain language model which is linearly interpolated with the out-of-domain language model. Then translation hypotheses are re-created using the interpolated language model, before they build and run the queries and generate the in-domain language model. They achieve their best results using query models that incorporate additional structure in the queries.

Wu et al. (2008) use linear interpolation of language models as well as of translation models. However, instead of a given bilingual in-domain corpus, they employ an in-domain word dictionary for adaptation. They treat the dictionary as a small in-domain phrase table or as data for in-domain translation model training. In-domain and out-of-domain phrase tables are combined during decoding. Either each phrase table is used as factor in the log-linear...
translation model, or both are linearly interpolated similar to the language models. Their intermediate results suggested that the log-linear approach works better.

Koehn and Schroeder (2007) have a similar arrangement. Their simplest phrase table adaptation setup is to combine in-domain and out-of-domain bilingual corpora before training. A more advanced way is to create two separate phrase tables, which are combined using factored translation models (Koehn and Hoang, 2007). They create an adapted language model in different ways, either using only the in-domain LM, linearly interpolating it with the out-of-domain LM, or using both as separate features in the log-linear translation model.

A quite different approach to domain adaptation is automatic post-editing (APE). In manual translation, a translator who corrects output from an MT system does post-editing. In automatic post-editing, manual corrections are used to train a system that automatically corrects the output of the original MT system. In such way, the post-edit system should learn to relieve the editors of repeatedly fixing the same mistakes made by the MT system. We experiment with this by training one SMT system to correct translations made by another SMT system.

Isabelle et al. (2007) improve PORTAGE, a RBMT system, by the use of SMT as post-processing step. A bilingual corpus is constructed using the RBMT output translations as source text and the post-editor reference translations as target text. This corpus is used for SMT model training. In this setup, two translation steps are performed: the source text is translated by the RBMT to intermediate target language, which is translated by the APE layer to correct target language text. This process can be used to easily customize the RBMT system, or to adapt it to a specific domain. In their experiments, Isabelle et al. (2007) report results for a small APE-training corpus (< 500k words) of human corrections. The system yields almost the same results in BLEU score, as an RBMT system customized with 18000 manual entries. With an increase of APE training data, the overall quality improvement stagnates. The improvements seem to be limited by the output quality of the RBMT system.

Simard et al. (2007) report similar experiments using the PORTAGE system. Dugast et al. (2007) worked on improving the SYSTRAN RBMT system by statistical post-editing. They work with the English–French language pair and confirm good results by automatic evaluation as well as linguistic analysis. The SPE layer mostly improves local word choice, degrades morphological accuracy and does not affect long-distance reordering (which the RBMT does well).

In similar work, Díaz de Ilarraza et al. (2008) concentrated on the Spanish–Basque language pair, where little bilingual material is available. They use the open source RBMT system Matxin (Alegria et al., 2007). As Basque is a morphologically rich language, each word in the source corpus was replaced by its stem and additional morphological tags. Tests with this morpheme-based SMT system show significant improvements in NIST, WER and PER scores over the word-based SMT system (except for BLEU scores, which are worse). Their results are consistent with other research for a restricted domain corpus. However, for a general domain corpus the plain SMT system outperforms the combination of RBMT system with SPE module.

3 Data and Methods

3.1 Data

Our baseline translation system is trained on a reasonably large amount of out-of-domain parallel data to get a state-of-the-art SMT system. We have only little parallel in-domain data and have no specific monolingual in-domain target language corpus for language model training. The amount of parallel data could be alleviated by considering some other language pair than Finnish–English, but we are interested in considering Finnish, as it is the local language and the reported challenges in translating to and from Finnish with statistical methods (Koehn, 2005). Our experiments investigate the case of starting to adapt translation system with very little in-domain data.

The Europarl (denoted as ‘ep’) corpus Koehn (2005) was our out-of-domain corpus for training the baseline translation (TM), reordering (RM) and language (LM) models. We created a small in-domain news corpus (denoted as ‘il’) for domain adaptation and evaluation. For the adaptation methods we concatenated the two corpora (denoted as ‘ep+il’) or trained separate models that were combined in a log-linear framework (denoted as ‘ep,il’) or linearly interpolated (denoted as ‘ep*il’). For post-edit models we paired the English output translations by the baseline system with the English reference translation for those sentences (corpus denoted as ‘pec’). The corpora were preprocessed with the standard Moses scripts, included lowercasing and tokenization.
3.1.1 Europarl Parallel Corpus

The baseline models were generated from the Europarl corpus version 2 Koehn (2005), which is a widely used parallel corpus in SMT research. The corpus is freely available and based on the web versions of the European Parliament proceedings from April 1996 to September 2003 in eleven languages.

For the experiments in this paper, we selected the English–Finnish data. It contains the proceedings data from January 1997 to September 2003 with a total of 0.8 million sentence pairs after standard preprocessing.

3.1.2 Iltalehti Parallel Corpus

The monolingual Finnish in-domain corpus was extracted from the web version of Iltalehti, a Finnish daily tabloid newspaper. Sentence length was limited to minimum of 3 and maximum of 12 words. Sequences shorter than 3 words were not considered proper sentences and sequences longer than 11 words were considered too complex and laborious to manually evaluate and correct.

The extracted Finnish sentences were translated into English by volunteers using a web-based application. The created small in-domain parallel corpus consisted of 1076 sentences.

3.2 Baseline Translation System

All experiments were conducted with Moses Koehn et al. (2007), an open source, state of the art statistical machine translation system. During decoding we used the default settings except for a translation table limit of 20, word penalty of -1 and a distortion limit of 6. We used the default reordering model (msd-bidirectional-fe).

The Finnish–English baseline models were trained with Europarl corpus (Koehn, 2005). The 4-gram language models used in all experiments were created using the SRILM toolkit Stolcke (2002) with Kneser-Ney smoothing.

3.3 Adaptation Methods

Here we provide brief descriptions of the adaptation methods in our experiments. A short identifier for each experiment is given in parenthesis.

Language model adaptation (L_n) only modifies the language model component of the system, whereas translation model adaptation (I_n/C_n) modifies models responsible for translation. The post-edit adaptation (P_n) does not change the baseline model (B) but builds a new translation system that takes as input the output of the baseline system. Table 1 describes the different data sets, models and methods.

3.3.1 Language Model Adaptation

We experiment with two different approaches: one new model based on all data (L2) and linear (L3) and log-linear (L1) model interpolation between baseline and in-domain models. We provide comparison with results with only in-domain (L4) and baseline language model (B).

We create linearly interpolated LMs with the SRILM toolkit and use Moses for log-linear combination (Equation 2) of LMs by using them as distinct features.

3.3.2 Translation Model Adaptation

Similarly to LM adaptation, we compare the baseline system to a new translation model that is trained on combined baseline and in-domain corpora (C_n) and a log-linear combination between separately trained baseline and in-domain models (I_n).

We show results with three different language models: a baseline language model (C1/I1), a LM trained on both corpora (C2/I2) and linear interpolation between baseline and in-domain LMs (C3/I3).

3.3.3 Post-edit Domain Adaptation

The automatic post-edit domain adaptation method does not work in parallel with the baseline model as in model interpolation, but the baseline and the APE models are in a sequence. Therefore, the approach is feasible even if the baseline system details cannot be accessed or modified. This method does not translate between two different languages, but rather tries to correct the baseline system output to match the reference translations.

As in the translation model experiments, we show results with three different language models: a baseline language model (P1), a LM trained on both corpora (P2) and linear interpolation between baseline and in-domain LMs (P3).

3.4 Evaluation

We evaluate translation quality using the BLEU score measure (Papineni et al., 2002), which is commonly used in MT research, although it has received much
The basic idea of BLEU is to reward closeness to one of the human reference translations, using modified unigram precision. The precision is determined by the weighted overlap of n-grams between candidate and reference translations for \( n = 1, \ldots, 4 \). The closeness between candidate and reference is given by the final score between 0 and 1.

Given our small bilingual in-domain corpus, it is hard to obtain a representative sample and the statistical significance of our results could be questioned. Therefore, a combination of 10-fold cross validation and bootstrap resampling of training data was used. Efron and Tibshirani (1986) used cross-validation to improve statistical accuracy. While not assuming any specific distribution for the data, bootstrap resampling has earlier been applied for significance tests in machine translation (Koehn, 2004; Zhang et al., 2004).

Due to the very small in-domain adaptation corpus, we did not create a separate tuning data set as this would have reduced the available data for training too much. However, default parameters could be unfortunate in a way that the comparison between different systems is unfair. Therefore, a combination of 10-fold cross validation and bootstrap resampling of training data was used. Cross-validation enlarges the variability of the training data and bootstrap resampling improves statistical accuracy for test set evaluation, while not assuming any specific distribution for the data. Bootstrap resampling has earlier been applied for significance tests in machine translation (Koehn, 2004; Zhang et al., 2004).

The training sets were used to train translation models, reordering models, and language models. The testing sets were used during automatic evaluation and for trying to obtain reasonable interpolation weights between in-domain and out-of-domain models.

Target language translations of each cross-validation model were resampled with replacement to form 1 000 new sets of 100 sentence test corpora. The test set for each cross-validation fold was resampled and the BLEU score for each of these 10 · 1 000 test sets evaluated. These BLEU scores were then combined to determine the bootstrap confidence interval and mean estimates.

For system ranking we perform a pairwise comparison as described in Zhang et al. (2004), but use a one sided 95% interval. The method can be described as first calculating the difference between each pair of samples and subsequently verifying if 95% of the differences are smaller than zero for any one of the participating systems. If the condition is met, the score difference is significant at the 95% level. To our knowledge, this procedure has not been widely applied in MT research yet, therefore we also rank the systems using the Wilcoxon signed rank test Wilcoxon (1945).

### 4 Results

Evaluation for all systems are given in Table 2 which shows the data and models used for training and test sets.

<table>
<thead>
<tr>
<th>Id</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ep</td>
<td>Europarl (Finnish,English) corpus</td>
</tr>
<tr>
<td>il</td>
<td>Iltalehti (Finnish,English) corpus</td>
</tr>
<tr>
<td>pec</td>
<td>Post-edit corrections (English,English)</td>
</tr>
<tr>
<td>ep+il</td>
<td>One model trained on combined corpora</td>
</tr>
<tr>
<td>ep,il</td>
<td>Log-linear combination of models</td>
</tr>
<tr>
<td>ep*il</td>
<td>Linear interpolation of models</td>
</tr>
</tbody>
</table>

Table 1: Description of data, models, and system descriptions.

Our results with language model adaptation (L\(n\)) are not in line with existing research, as we were only able to get improved BLEU scores with a new model trained on all data (L2). Koehn and Schroeder (2007) try similar experiments as our Ln, although with a significantly larger in-domain corpus. Compared to the baseline, they report improvements in each of the LM adaptation methods, where the simple combination of corpora (comparable with L2) performs worse than the other methods. For our LM adaptation methods, L2 performs best. For linear LM interpolation, we tried different weights using LM perplexity as a guide. However, the weight giving the lowest perplexity (0.5) did not result in the best translation score; similar scores were instead achieved for weights between 0.5 to 0.9.

Linear LM interpolation (L3) outperformed log-linear LM combination (L1), which agrees with the results in Wu et al. (2008). We tried different weights for the two LMs in the log-linear LM combination (L1), but additional in-domain LM weight only seemed to degrade translation performance.

All translation adaptation methods (CJ/IJ/PN) outperformed the baseline system. Language model, however, did influence the results as interpolated LMs (ep*il) produced lower scores than the baseline model (ep), whereas the retrained model (ep+il) was always the best. The only exception was the post-edit adaptation (PN), where also linear interpolation of LMs outperformed the baseline LM.
<table>
<thead>
<tr>
<th>Id</th>
<th>Description</th>
<th>Data</th>
<th>Training</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TM</td>
<td>RM</td>
<td>LM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mean</td>
<td>interval</td>
<td>interval</td>
</tr>
<tr>
<td>B</td>
<td>baseline, no adaptation</td>
<td>ep</td>
<td>ep</td>
<td>ep</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16.49</td>
<td>[15.08, 17.79]</td>
<td>[12.64, 20.38]</td>
</tr>
<tr>
<td>L1</td>
<td>log-linear LM combination</td>
<td>ep</td>
<td>ep, il</td>
<td>ep</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20.92</td>
<td>[11.89, 14.68]</td>
<td>[9.91, 16.88]</td>
</tr>
<tr>
<td>L2</td>
<td>combined corpus LM</td>
<td>ep</td>
<td>ep+il</td>
<td>ep+il</td>
</tr>
<tr>
<td></td>
<td></td>
<td>19.79</td>
<td>[13.79, 15.93]</td>
<td>[11.42, 18.45]</td>
</tr>
<tr>
<td>L3</td>
<td>linear LM interpolation</td>
<td>ep</td>
<td>ep</td>
<td>ep+il</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20.29</td>
<td>[9.45, 12.08]</td>
<td>[7.808, 13.87]</td>
</tr>
<tr>
<td>C1</td>
<td>combined corpus TM/RM</td>
<td>ep+il</td>
<td>ep+il ep+il</td>
<td>ep+il ep+il</td>
</tr>
<tr>
<td></td>
<td></td>
<td>48.92</td>
<td>[19.58, 23.23]</td>
<td>[16.79, 26.32]</td>
</tr>
<tr>
<td>C2</td>
<td>+combined corpus LM</td>
<td>ep+il</td>
<td>ep+il ep+il</td>
<td>ep+il ep+il</td>
</tr>
<tr>
<td></td>
<td></td>
<td>55.70</td>
<td>[20.55, 24.28]</td>
<td>[17.50, 27.57]</td>
</tr>
<tr>
<td>C3</td>
<td>+linear LM interpolation</td>
<td>ep+il</td>
<td>ep+il ep+il</td>
<td>ep+il ep*il</td>
</tr>
<tr>
<td></td>
<td></td>
<td>56.19</td>
<td>[19.73, 22.73]</td>
<td>[16.57, 26.20]</td>
</tr>
<tr>
<td>I1</td>
<td>log-linear TM combination</td>
<td>ep</td>
<td>ep</td>
<td>ep</td>
</tr>
<tr>
<td></td>
<td></td>
<td>62.92</td>
<td>[21.87, 25.64]</td>
<td>[18.77, 29.04]</td>
</tr>
<tr>
<td>I2</td>
<td>+combined corpus LM</td>
<td>ep, il</td>
<td>ep, il ep, il</td>
<td>ep, il ep, il</td>
</tr>
<tr>
<td></td>
<td></td>
<td>68.98</td>
<td>[22.49, 27.03]</td>
<td>[19.52, 30.39]</td>
</tr>
<tr>
<td>I3</td>
<td>+linear LM interpolation</td>
<td>ep, il</td>
<td>ep, il ep, il</td>
<td>ep, il ep, il</td>
</tr>
<tr>
<td></td>
<td></td>
<td>69.89</td>
<td>[21.41, 25.44]</td>
<td>[18.28, 29.08]</td>
</tr>
<tr>
<td>P1</td>
<td>post-edit TM/RM</td>
<td>pec</td>
<td>pec</td>
<td>pec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>61.02</td>
<td>[22.35, 25.75]</td>
<td>[18.47, 30.01]</td>
</tr>
<tr>
<td>P2</td>
<td>+combined corpus LM</td>
<td>pec</td>
<td>pec</td>
<td>pec+il</td>
</tr>
<tr>
<td></td>
<td></td>
<td>61.23</td>
<td>[21.81, 25.16]</td>
<td>[17.99, 29.35]</td>
</tr>
<tr>
<td>P3</td>
<td>+linear LM interpolation</td>
<td>pec</td>
<td>pec</td>
<td>pec+il</td>
</tr>
<tr>
<td></td>
<td></td>
<td>61.23</td>
<td>[21.81, 25.16]</td>
<td>[17.99, 29.35]</td>
</tr>
</tbody>
</table>

A comparison of the best systems (B/L2/C2/I2/P2) in each method is shown in Figure 1 as a histogram of the BLEU scores from the bootstrap resampling sets. This result was used to rank the adaptation methods, which gave the result given in Table 3.  

### Table 3: A system ranking of the best systems of each method.

<table>
<thead>
<tr>
<th>Id</th>
<th>Description</th>
<th>Data</th>
<th>Training</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TM</td>
<td>RM</td>
<td>LM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mean</td>
<td>interval</td>
<td>interval</td>
</tr>
<tr>
<td>B</td>
<td>baseline, no adaptation</td>
<td>ep</td>
<td>ep</td>
<td>ep</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16.49</td>
<td>[15.08, 17.79]</td>
<td>[12.64, 20.38]</td>
</tr>
<tr>
<td>L1</td>
<td>log-linear LM combination</td>
<td>ep</td>
<td>ep, il</td>
<td>ep</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20.92</td>
<td>[11.89, 14.68]</td>
<td>[9.91, 16.88]</td>
</tr>
<tr>
<td>L2</td>
<td>combined corpus LM</td>
<td>ep</td>
<td>ep+il</td>
<td>ep+il</td>
</tr>
<tr>
<td></td>
<td></td>
<td>19.79</td>
<td>[13.79, 15.93]</td>
<td>[11.42, 18.45]</td>
</tr>
<tr>
<td>L3</td>
<td>linear LM interpolation</td>
<td>ep</td>
<td>ep+il</td>
<td>ep+il ep*il</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20.29</td>
<td>[9.45, 12.08]</td>
<td>[7.808, 13.87]</td>
</tr>
<tr>
<td>C1</td>
<td>combined corpus TM/RM</td>
<td>ep+il</td>
<td>ep+il ep+il</td>
<td>ep+il ep+il</td>
</tr>
<tr>
<td></td>
<td></td>
<td>48.92</td>
<td>[19.58, 23.23]</td>
<td>[16.79, 26.32]</td>
</tr>
<tr>
<td>C2</td>
<td>+combined corpus LM</td>
<td>ep+il</td>
<td>ep+il ep+il</td>
<td>ep+il ep+il</td>
</tr>
<tr>
<td></td>
<td></td>
<td>55.70</td>
<td>[20.55, 24.28]</td>
<td>[17.50, 27.57]</td>
</tr>
<tr>
<td>C3</td>
<td>+linear LM interpolation</td>
<td>ep+il</td>
<td>ep+il ep+il</td>
<td>ep+il ep*il</td>
</tr>
<tr>
<td></td>
<td></td>
<td>56.19</td>
<td>[19.73, 22.73]</td>
<td>[16.57, 26.20]</td>
</tr>
<tr>
<td>I1</td>
<td>log-linear TM combination</td>
<td>ep</td>
<td>ep</td>
<td>ep</td>
</tr>
<tr>
<td></td>
<td></td>
<td>62.92</td>
<td>[21.87, 25.64]</td>
<td>[18.77, 29.04]</td>
</tr>
<tr>
<td>I2</td>
<td>+combined corpus LM</td>
<td>ep, il</td>
<td>ep, il ep, il</td>
<td>ep, il ep, il</td>
</tr>
<tr>
<td></td>
<td></td>
<td>68.98</td>
<td>[22.49, 27.03]</td>
<td>[19.52, 30.39]</td>
</tr>
<tr>
<td>I3</td>
<td>+linear LM interpolation</td>
<td>ep, il</td>
<td>ep, il ep, il</td>
<td>ep, il ep, il</td>
</tr>
<tr>
<td></td>
<td></td>
<td>69.89</td>
<td>[21.41, 25.44]</td>
<td>[18.28, 29.08]</td>
</tr>
<tr>
<td>P1</td>
<td>post-edit TM/RM</td>
<td>pec</td>
<td>pec</td>
<td>pec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>61.02</td>
<td>[22.35, 25.75]</td>
<td>[18.47, 30.01]</td>
</tr>
<tr>
<td>P2</td>
<td>+combined corpus LM</td>
<td>pec</td>
<td>pec</td>
<td>pec+il</td>
</tr>
<tr>
<td></td>
<td></td>
<td>61.23</td>
<td>[21.81, 25.16]</td>
<td>[17.99, 29.35]</td>
</tr>
<tr>
<td>P3</td>
<td>+linear LM interpolation</td>
<td>pec</td>
<td>pec</td>
<td>pec+il</td>
</tr>
<tr>
<td></td>
<td></td>
<td>61.23</td>
<td>[21.81, 25.16]</td>
<td>[17.99, 29.35]</td>
</tr>
</tbody>
</table>

5 Conclusions and Discussion

This paper experimented with a statistical machine translation framework and four different domain adaptation methods from the baseline system trained on the Finnish–English part of the Europarl corpus to a news domain. The in-domain news corpus was a very small parallel corpus collected by the authors. The results show that the adaptation methods can sig-

---

2 All rankings use a significance level of 95%.

3 Based on the assumption that the data is t-distributed.
significantly improve translation quality measured with the BLEU score, even when the in-domain training corpus is very small. Our results suggest that language model adaptation combined with translation model adaptation methods or post-editing methods produces the best results. Language model adaptation methods by itself may not always improve the results.

We were not able to show a clear ranking of the adaptation methods. However, the choice of the appropriate method might depend not only on the improvements in translation quality but also on other performance measures, such as training time, model size and translation time.

Due to the small size of the in-domain parallel corpus, parameters of each system were not fully optimized. Some of our results deviate from existing research, especially regarding the language model interpolation, which we suspect to be a result of the very small in-domain corpus size. In further work, we believe that combining the methodologically different adaptation methods would produce even greater improvements in translation quality.

References


GArphics – Applying genetic algorithm for generating graphics

Anssi Jäntti
University of Vaasa, Department of Electrical and Energy Engineering
P.O. Box 700, FI-65101 Vaasa, Finland
anssi.jantti@iki.fi

Jarmo T. Alander
University of Vaasa
jarmo.alander@uwasa.fi

Abstract
This paper deals with computer program called GArphics, how does it work, how was it implemented and why. The program generates a random set of mathematical functions and draws images based on those functions. It then uses an interactive genetic algorithm to generate more variations of images. The program and the source code are published under the GNU General Public License (version 2) and are freely available for download at http://garphics.codeplex.com/

1 Introduction
The development of this program began as an exercise for the Soft computing course AUTO2050, arranged by the Department of Electrical Engineering and Automation of the University of Vaasa on the year 2009. The task, given by professor Jarmo Alander, was to develop a program which would generate graphics with genetic programming.

The principle on GArphics is that first a function
\[ z = f(x, y) \]  
(1)
is randomly generated and then the value \( z \) is calculated for every pixel \((x, y)\) on the target image. Values \( z \) are scaled to 24-bit integers, which are then further converted to RGB-colours. In essence, a graph of the function is drawn.

On the program, user has the ability mutate and cross the functions and in this way search for more pleasing images with genetic algorithm. The generated functions can be saved to XML-files which can later be loaded back for more processing. It is also possible to modify most of the parameters affecting the genetic algorithm, save pictures to bitmaps and generate animations based on the functions.

2 Related work
Karl Sims presented principles for applying genetic algorithms for a very similar problem as it has been done here with GArphics. Karl Sims (1991) used a set of Lisp functions as a basis for generating the images. GArphics has been heavily influenced and inspired by the works of Karl Sims. GArphics uses a tree structure to store the mathematical functions and Hewgill and Ross (2004) have used very a similar structure when creating 3D textures with genetic programming. Muni et al. (2006) used genetic programming to create fashion designs and their results are very similar to some of the pictures which GArphics produces.

For more references on genetic algorithms and genetic programming in computer graphics and art in general, see for example the indexed bibliography of genetic algorithms in arts and music (Alander 2010).

3 Implementation
This section describes in detail the implementation of the program and its functionality. The program is published under the GNU General Public License (version 2) and the source code has been made freely available for anyone to use and study in more detail. The source code can be downloaded from the programs webpage at CodePlex (Microsoft Corporation, Redmond, Washington, USA): http://garphics.codeplex.com/

GArphics has been written with C#-programming language, using Visual Studio 2008 (Microsoft Corporation, Redmond, Washington, USA). GArphics requires the .NET Framework 2.0 to run. The main window of the program is show in Figure 1.

3.1 Operators
Mathematical operators used in the program are the following: +, −, *, /, bitwise AND, bitwise OR, MOD, ABS, SIN, COS, TAN, ACOS,
ASIN, ATAN, ATAN2, COSH, EXP, LOG, LOG10, MAX, MIN, POW, SINH, SQRT and TANH. Operators are mostly from the standard C# System.Math-library. The calculation of modulo and division operators have been given an additional check to prevent the usage of zero-divisor.

Implementation of the AND- and OR-operators is however a bit more complicated, since they cannot be used with floating-point numbers; Operands are scaled to their biggest possible integer-format, by multiplying them with the value

\[ f_m = 10^{\lceil \text{max}(|x|,|y|) \rceil}, \]

where \( x \) and \( y \) are the operands to be used with the AND- and OR-operators. If the operands were simply converted to integers with the System.Convert.ToInt32()-function, a lot of the precision would be lost especially on smaller values, because all of the values below 1 would be truncated to 0 and this would in turn cause a lot of the pictures to show simply as black.

If, after all, there is an overflow or any other unexpected exception caused by the calculation, a Fail-Value is returned as a result. The value is set to zero by default, but this can be easily changed from the Settings-menu.

Figure 1: The main window of GArphics version 0.9, with the built-in examples loaded.
3.2 Binary tree structure

GArphics uses a full binary tree structure to store the mathematical equations. This structure was chosen to simplify the implementation of the program and the usage of the genetic algorithm. However some of the mathematical operators can only be used for one operand and in this case some of the branches of the tree might not have any effect on the constructed image. These inactive genes may however become active after mutations and crossovers. Inactive genes have been circled with red on the Figure 2.

Figure 2 shows an example of a formula tree and the corresponding image generated by it when $-1 < x < 1, -1 < y < 1$. Function represented on the tree structure can also be written as

$$f(x, y) = 0, 4905431 \& |\cos(y \mod x)|.$$  \hspace{1cm} (3)

Currently inactive genes are circled with red.

3.3 Rendering properties

In addition to the operators and operands, the object model defining the tree structure is used to store the interval inside which the picture is to be drawn and information about colour-filtering.

By default, the images are drawn on the interval $-1 < x < 1, -1 < y < 1$, but this can be easily changed when running the program. The program also has an option to let the genetic algorithm to search for the best interval.

Colour-filters can be used to disable the individual colour-channels by setting them to some constant value. The genetic algorithm can be used to search for the best colour-filter combination and in addition to that, user can manipulate the filters manually.

Furthermore the object model of the tree-structure used in the program has functionality for calculating the values, of the mathematical function the tree stores, on given coordinates. The tree-structure also has the methods needed for the genetic algorithm, described in more detail on the Section 3.5. These methods include the creation of random tree, mutating tree and combining two trees.
3.4 Creating the images

This section describes in more detail how the pictures are constructed from the tree-structures.

When the tree structure is to be converted to image, a table of double-precision floating point numbers, with size of the target images resolution, is created to the computer’s memory. The table is then filled by calculating the values of the function (described on the tree) on the coordinates which correspond with the pixels on the target image. So if we have a target image with $4 \times 4$ resolution and the tree is to be drawn on interval $-1 < x < 1$, $-1 < y < 1$, the values are calculated as shown on the Figure 3.

When all of the values have been calculated, they must be converted to RGB-colours. Simplest way to do this would be to multiply the values with 255 and thus the result would be a greyscale image (as show on Figure 4). However all the values below 0 would result as black and values above 1 would result as white and because of this the simplest system would not be a feasible solution with more complex functions.

Another method to convert the values to picture, also implemented on GArphics, is to scale all values to 24-bit integer, so that the lowest value becomes 0 and the highest value becomes 16777215. These values are then easy to convert to 24-bit colour. This can be done for example by taking the red colour channel from the lowest 8 bytes, green channel from the middle bytes and blue channel from the highest 8 bytes.

Figure 4 shows the result of constructing greyscale images from the following three simple functions. First image is created with the function

$$ f(x, y) = x $$

and thus results as a grading which goes from black to white. Second image has been created with the bitwise AND

$$ f(x, y) = x \& y $$

and the last with function

$$ f(x, y) = \cos(x) \mod y. $$

3.5 Genetic algorithm

Typically genetic algorithms (Alander 2006) have four primary parts or steps; Initializing the population, selecting the best members, changing some parts of the members (with random mutations or by combining parts of the old members, thus creating new ones) and repeating the process with new population.

On this program the progress of the genetic algorithm is controlled by the user. User can decide which of the images is the best or most pleasing. User can choose to produce variations on certain picture by mutating it or combining two pictures. User can also decide when the desirable image has been achieved and end the progress or the genetic algorithm, save the picture, create animation or start a new round.

3.5.1 Mutations

When user chooses to perform mutations on one of the members of the current population in GArphics, user first selects the targeted member and then clicks the "Mutate"-button. Program performs mutations on the selected member and generates a total of 19 children. The parent is automatically passed on to the next round.

The mutations are performed by first randomly selecting one of the parameters of the selected tree and then changing it randomly. If the selected parameter is for example one of the tree-nodes, the operator or operand stored in the node can be changed or a whole new branch of a tree might be created. Other possible parameters include the values defining the colour-filters and ranges inside which the picture is drawn.

Because minor changes in the trees do not usually result in major visible changes in the generated pictures, GArphics can be set to automatically perform several mutations during one round. Default value is one to five mutations per generated child, but this and all other probabilities relating to the mutations can be changed by the user. The addition probabilities include the changing of the operators, operands, colour-filters, coordinate system and the probability or creating new branches.

3.5.2 Crossovers

In GArphics the crossovers are done by combining the one-point crossover and uniform crossover. The
operator trees of the parents are combined by selecting one node from each parent and their places. The other tree-properties are combined uniformly. 18 children are created from two selected parents on each crossover-round.

### 3.6 Animations

In addition to the images, the program can be used to create animations based on the produced images. The animation is created by drawing a set of images based on the selected equation and by tightening the interval on which the image is drawn. The lower bounds on X- and Y-axes are increased while the upper bounds are decreased. The increasing is done with the formula

\[ a_{n+1} = a_n + \frac{|a_n|}{b} \]  

(7)

and decreasing with

\[ a_{n+1} = a_n - \frac{|a_n|}{b} \]  

(8)

where \( a_n \) is the current limit, \( a_{n+1} \) is the new limit and \( b \) is a value (from 2 to 100) given by the user. So in essence the program zooms in the image.

User can also specify how many frames should be calculated for the animation and how fast the frames should be changed. The amount of tightening the interval combined with the number of frames, decides how deep in to the image is zoomed. The number of frames combined with the frame rate in turn decides how long the final animation will be.

The created animations can be saved as a GIF-animation and individual bitmaps. The saved animations can then later easily be converted into some other video-format with separate tools, if required.

### 4 Conclusions and future

The actual purpose of this project was to complete the required exercise, to see if it could actually be done and generally explore the realms of genetic algorithms. This project was so-called "blue skies research".

The initial task for the exercise was just to create a small and simple program which would use a genetic algorithm to create images. However as the curiosity peaked, the development of the program took a course of its own. The final result of the project was a highly configurable interactive genetic algorithm, with the ability to save and load to XML and to generate animations. And if time permits, even more features might be implemented (see Section 4.1 Future work).

![Figure 5: Example 6 from the GArphics](image)

Even when there initially were no planned applications for the program, some possible applications came up during the development. This program could for example be used to generate pictures for testing image processing algorithms (Mantere 2007), to generate computer background-pictures, screensavers, textures for games, or simply just as a way to demonstrate the possibilities of genetic algorithms in general.

The images shown in Figures 5, 6 and 7 were all generated with the program. The functions, which these images are based on, are available as examples on the program itself.

### 4.1 Future work

Parallel computing is already used when the smaller images are drawn in the programs main-view, so that a separate thread is created for drawing each of the images and the number of threads can be changed from the Settings-menu. However calculating the bigger images and especially full-window animations would far greatly benefit from parallel processing if for example the images could be divided into sub-blocks which in turn would be calculated on different threads. This is something that might be implemented on future versions.

The program might be able to create more complex and better images if some image-processing functions, like dilation, erosion, blur and sharpen were added. There could also be a feature in the program which would allow user to modify the mathematical
functions manually. Implementing different ways of converting the calculated values to colours and usage different colour spaces might yield interesting results.

Animations are currently drawn by narrowing the interval inside which the pictures are drawn. Animations could as well be created by widening the interval or shifting it along the X- and Y-axis. The movement could even be done with some trigonometric functions like sine and cosine and thus result would be smooth curves. Different kind of animations could be generated with genetic algorithm, but this would require a lot of processing power, since even the calculation of one animation takes relatively long time.

4.1.1 3D-images

One of the most requested features, which have come up when the functionality of the program has been demonstrated to various people, has been the ability to create 3D-images. This could be done for example by adding a third coordinate (in addition to X and Y) to the functions which are currently processed by the program. This would however result in kind of coloured clouds and the viewing of this kind of images on conventional displays would be difficult.

Another possible way of using the program to create 3D-images would be to use the colour value of each pixel to define the height of the 3D-point. This method would in turn result in coloured landscapes. Some other value, like contrast, brightness, gamma or some individual colour-channel, might also be used to define the heights instead of the colour values.

4.1.2 Image filtering

GArphics currently uses a fully interactive genetic algorithm, which relies completely on the user of the program to decide which images are better than others. Muni (2006) introduced some methods of programmatically evaluating the images and calculating fitness functions and thus reducing the burden of the users.

It would be interesting a filter which would be able to filter out those images which only show as black or single colour. This functionality could be in turn developed even further to a point where the program could for example use the difference of contrast and gamma of neighbouring pixels to try evaluate if the image would appear as pleasing to the user.

Acknowledgements

Thanks to all the teachers who have taught me programming, mainly Olli Snellman and Aki Vainio. Special thanks to Mika Ruohonen for all the help with this paper. And finally thanks everyone else who supported or helped me with this paper in any way.

References

Jarmo T. Alander. Possibilities of genetic algorithms (Report 97-4, in Finnish). Vaasa, Finland: Department of Electrical and En-


A synchronized system concept and a reference implementation for a robot dog

Csaba Kertész
Vincit Oy
Hermiankatu 3 A, 33720 Tampere, Finland
csaba.kertesz@ieee.org

Abstract

Couple of different system concepts have been emerged in the past decades in order to control the robots efficiently. Usually, the robots have resource constraints on processor, memory and power, therefore, the engineers must optimize the algorithms for these conditions. This paper proposes a concept what can be used to design a synchronized system and utilize the increasing computational power of the computers in these days to extend the capabilities of the robots. A reference implementation is presented for the Sony AIBO robot dog, but the concept can be applied to design other kind of systems, which rely on synchronized operations.

1 Introduction

The software of the robots can be run in three basic ways. First, the algorithms can be implemented in hardware components, which offers the fastest execution, though the debugging and bug fixing can be time-consuming. The walking strategy of the hexapod FOBOT (Vámossy et al., 2004) was programmed in PIC microcontrollers to manipulate the servos with low-latency. Building the integrated circuit and the implementation of the strategy was a long process.

An operating system on the robot is more flexible, and after writing the algorithms in programming languages, the sources can be compiled and transferred for on-board execution. The program runs on the robot, the execution is slower than the hardware-accelerated and the development can be a bit complex to debug the algorithms. The Willow Garage's robot platform (PR2) has a high computational potential with two Quad-Core processors in two on-board servers. Maitin-Shepard et al. (2010) concentrated on the towel folding problem with this robot and achieved notable results.

When the computational resources are restricted on the robot, the sensor data (camera image, IR) can be transferred to a remote computer and processed there. Debugging and maintaining the software stack on the computer may be an easier job, but an obvious disadvantage is the high latency of the responses back to the robot. Geusebroek and Seinstra (2005) streamed the camera image of AIBO and used grids of computers around the world for object recognition.

The projects seldom follow these clean principles in the real life, but a mixture is applied for each case. The Universal Robotic Body Interface (URBI) provides both the on-board and the remote execution with the same programming interface (Baillie, 2005). Remote objects can be plugged in on demand for debugging purposes or a distant computer can be used for expensive computation. The URBI runs on a variety of robots like AIBO or Nao, which is used in the Humanoid League of the RoboCup nowadays (Hester et al., 2010).

The advanced robots in the research are expensive, therefore, the development should minimize the run-time of these robots. The simulation environments, like Webots (Hohl et al. 2006), can replace the robots in a certain amount of development time, but the imitations of the real environments are not perfect. The Webots can save the 3D animation of the simulation, however, it can not replay the same situation with the program of the robot if the physics of the simulated environment contains variable conditions; the butterfly effect can make impossible to reproduce these situations. Various challenges can extend the development time with a robot: abnormal termination of the on-board software (e.g. segmentation fault), wrong interpretation of the sensor input (e.g. false object detections) or problems with the locomotion of the robot.

There is no perfect solution, but the new system concept of this paper offers help to overcome some difficulties with robots. For instance, any change in
the source codes can lead to unexpected effects in the behaviour of the robot. The concept can record the incoming data in the system and simulate the situation later. This approach can be used to test the changes in the software and mistakes in the development can be fixed without running the robot. Switching the obsolete test records to new can be done by the simple recording interface API in the concept.

The author's purpose with the proposal fits into the remote execution for robots with resource constraints, but it can also be applied in on-board software stack. The usual frameworks are robot and technology dependent, but the proposal lets the developer to use the framework for any robot or other purposes need synchronous operations. On the other hand, the disadvantage of this approach is the lack of the robot interface, it must be implemented from scratch completely.

The proprietary programming environment of AIBO, the Open-R (Roci et al, 2004), offers platform dependent support for remote execution of the programs for the robot and the on-board run provides only a console via wireless connection for debug messages and a restricted backtrace function in the case of an abnormal termination on the robot. The concept in this paper has a chance to deploy the programs for the robot and the on-board run provides only a console via wireless connection for debug messages and a restricted backtrace function in the case of an abnormal termination on the robot.

Similar to SPQR-RDK (Farinelli et al, 2006), the elements can be added and removed from the working system of the concept thanks to the modularity of the elements as well as the data flows between the elements have a standardized form and the computations are synchronized what is an important feature of the neurons' data processing in the human brain (Schoppa, 2006).

The reference implementation, described after the concept, can utilize multiple processor cores or computers to increase the processing power and it uses the Urbiscript to control the body of the robot that helps the re-usability of the developed algorithms, because the URBI is available for multiple robots beyond AIBO. The next section details the concept.

## 2 Synchronized system concept

### 2.1 Overview

The system gets input data from outside and processes it. The internal behaviour of the system is (hopefully) deterministic and reproducible later by restoring its initial state and reloading the recorded incoming data. Acting as a scheduler, the synchronicity of the system components is provided by periodical beat signals, using the signal-slot method of the Qt (Molkentin, 2007). The basic definitions:

- **Surroundings**: Everything outside the system.
- **Inflow**: The raw data coming from the surroundings to the system.
- **Context**: It creates, maintains the low-level behaviour of the system and hides from the developers.
- **Normal element**: An entity of the system, which implements data processing functions. It may have input and output.
- **Source element**: The entry point of the inflow from the surroundings to modify the states of the internal components. It receives for example images, translates into a data model of the system and sends to the interested elements.
- **Feeding**: A normal or source element sends output to the interested parties.
- **Sink element**: The sinks receive data from the other elements and represent back to the developer (e.g. playing sounds on speakers). They have important role in the debugging and can have graphical interface to view data.
- **Debug window**: A special sink element with GUI to debug the circulating data in the system. It has a plug-in interface to interpret the future data models.
- **Element data**: The basis of the information exchange between the elements.
- **Data model**: The internal members of the data subclasses. The base data class does not contain variables, but its subclasses declare the new member variables to represent data in the system.
- **Input/output data**: An element data is an output of the sender, however, the same data is an input for the receivers. The sources do not have input, but they can provide any number of output. The sinks get the input data from other elements and they do not output. Any normal element can have any number of inputs and outputs.
- **Session**: One run of the system, interacting with the surroundings.
- **Session storage**: A storage to save the outputs of the source elements and replay a session later based on the records. A plug-in interface interprets the future data models for the storage.
- **Heart**: An entity of the system, which synchronizes the activities of the elements, the data exchange and the session storage with heart beating.
- **Heart beat**: One “tick” of the heart. During a beat, the elements have a certain amount of time for data processing.
Activities: The data processing functions of the elements during a heart beat. The activities mean inflow fetch for the sources, data manipulation for the normal elements and data representation for the sinks.

Components: The common phrase for the internal parts of the system: elements, session storage and heart.

Figure 1 shows the theoretical schema of the concept. The context controls each internal component, the heart synchronizes, the session storage saves or reloads a session and the debug window, as a special sink element, receives the data from the other elements in order to view for the developer. The following subsections detail the context and the components in the system.

2.2 Context

To build a system, an application should create the elements, an instance of the context and start the heart beating. Everything else is automated and hidden from the developer: the context internally creates and manages the heart, the debug window and the session storage. Note that the context does not destroy the elements, it should be done by the application eventually, similar to the creation of the elements.

When the elements are created, usually by declaration, in an application, their constructors register the element instances in the context automatically. Before the heart beating is being started, the elements are also registered by the context in the heart and the session storage is initialized.

To set the properties of the actual session (e.g. saving a session in a specific location on the hard drive), parameters can be passed to the context by the options of the application and there are several setter and getter functions in the public API of the context for the same purpose.

When an application is terminated in an abnormal way (e.g. segmentation fault), it is caught by the context and the session storage is tried to close to avoid data corruption (e.g invalid video file without index table).

2.3 Element data

In the synchronized data flow, the system components need to understand the model of the future data types, therefore, a base class is defined, as the basis of the data exchange. It has some important functions and variables that allow for the components (e.g. elements, session storage) to use data models derived from this class in the higher levels, without prior knowledge.

The solution is a string-to-pointer mapping of the members of the future data models, which must be added by the subclasses in a specific format: name, colon and type. For example, the mapping of a new integer variable called "Power" is "Power:int". The map of the strings and generic pointers to the data members is administrated in the base class.

The definition of the mapping is not vital for every new member variable though the omitted variables will not be visible for the components using the mapping to understand the data models. The assignment of the derived data types work for all, both mapped and non-mapped member variables.

2.4 Elements

The duty of the elements in the system is the data processing. The sources fetch the inflow from the surroundings and deliver to the interested parties. The normal elements can receive, process and send data to the others. The sinks have the capability to use a graphical, CLI or other sort of interface to represent some data back to the developer. The type of the element should be declared in its class constructor and kept permanent for the object lifetime.

The multiple instances of the element types have a unique ID (number) what is important to differentiate them. The numbers are set internally when the objects are created and they can be changed, though not encouraged, because the system components rely on (e.g. the session storage stores the output of the sources based on the element type and the ID). The numbering starts from 1 and it is incremented by one for each new instance.

The elements use the subclasses of the base data class for the data exchange. Each data flowing in the system must have a unique name, composed in the
form: name, colon, type. For instance, "Sensors:SensorData" means a data named "Sensors" with the type "SensorData" derived from the base data class. The inputs and the outputs must have corresponding member variables and declared name-to-pointer mappings to the variables in the constructors of the sender and the receiver elements.

By default, the inputs and the outputs of the elements are automatically connected by the context based on their names right before starting the heart beating; the elements are informed about the interested parties regarding a specific output. This step can be done manually, but not recommended. The actual feeding of the receiver elements is done in each beat behind the scenes.

Some inflows from the surroundings can not be available all cases, therefore, the sources have a temporary list of the actual outputs at the end of each heart beat. The sources get the raw data from the surroundings during a beat, turn into the data models of the outputs, fill the actual output list and inform the session storage to save the outputs if the session is being recorded. When the next beat starts, the heart commands the source elements to feed the registered elements with the available data, nevertheless, the implementation of the receivers should take into account that the input data is not always provided. The normal elements act in the similar way, they add a data to their actual output list if it is available.

The activities are not the same for each element type. The sources fetch and transform the inflow, the sinks update a graphical interface with new data or play sound on the speakers, however, the normal elements process their received input and send output to other elements. Let us assuming 10 Hz beating, there are 100 ms to perform actions by the elements and it is not enough for heavy computation. The elements should be designed in a proper way to avoid the blocking of the heart beating, because a beat ends in practice if every element sent a “finished” signal to the heart about the completed activities of the current beat. The heart warns the elements if the expected duration of the actual beat is up, but they are not forced to stop their activities.

At least two design principles can be used to resolve this issue:
- Divide the algorithm into smaller steps and calculate one smaller step during a beat.
- Use a separate thread/processor core for the element, do the calculations in the background in an idle continuously and get new input data/propagate the results through the heart beating.

Normally, the expectations for the source elements are the fast data fetch and providing the latest available inflow, therefore, the design of the sources should consider the fetching at the latest possible moment before the end of a beat, because the data sent to the elements in the begin of the beats is fetched during the previous beat. If the source knows the expected length of the current beat (e.g. 150 ms) and an estimation for the time needed to fetch the inflow (e.g. 40 ms), the activities of the source element can be scheduled to the latest possible time (150 ms - 40 ms = 110 ms). This delay can be refined for every beat.

2.5 Heart

A component, which synchronizes the activities of the elements and acts as a scheduler in the system. One smaller computational period corresponds to a beat in the terminology. A new beat is started if the previous is finished by all elements, however, the limit is not hard-coded for the duration. Any element can request a limit for the next beat, the heart receives that and sets a duration limit for the next beat. Using a beat frequency between 1-15 Hz is a good intention for a robot to get sensor data often and leave enough time for the data processing. If the beat duration time is up then the heart warns the elements to finish their activities, but the elements are not forced directly to stop their activities. The elements can also finish their activities earlier than the limit; a new beat begins immediately. The context can send an emergency signal to exit, but it is reserved for situations when the session dies with a segmentation fault or a similar signal.

The context registers the elements in the heart and the beating is started/stopped by the context as well. The activities of the sources are started in the first beat and all elements are active in the following beats. The reason is to let the sources to fetch the first portion of the inflow to feed the interested elements from the second beat. The beats inside a session are identified by a unique ID started from zero and it is always increased by 1 for the next tick. This ID has a vital role to record and play back the sessions.

On the Figure 2, the normal arrows follow the algorithmic steps during a beat. The heart sends a start signal for the elements and they progress with their tasks. When the data processing is done, the elements send “finished” signals to the heart and the iteration starts again. The dashed arrows indicate the data flow on the diagram, the session storage can load/save the output of the sources or for example a robot interface provides inflow (raw sensor data) for the sources. Eventually, the sources send the data to the receivers and normal elements can send output of their data processing from the previous beat. Note that the robot interface does not part of the core
system, it is not implemented here; showed solely for demonstration purposes.

Figure 2: The flow chart of a heart beat

The threads are used to parallelize functions to improve the performance (e.g. running independent calculations in separate threads/processor cores decreases the run-time), but the number of the processor cores determines the optimal performance. When a heavy calculation uses a core on maximum level, new computational threads can be run on other available cores, however, if all cores have full load, running a new thread does not boost the performance, because there is no free processor resource any more.

The Qt has a design restriction for the graphical widgets; they can be run only in the main thread of an application. For these two reasons above, the heart applies the following rules for the elements:
- The sink elements with GUI remain in the main thread.
- New, dedicated thread is created for each element with thread option enabled.
- A computational thread is created for all the remaining elements and they are moved there to avoid blocking the user interface.

2.6 Session storage

As mentioned in the previous subsections, the sessions can be saved by storing the output of the source elements in each beat and they can be reloaded later. This is a beneficial approach to simulate the system behaviour without active source elements, whose activities are skipped in that case.

The outputs are stored in a database or on the hard disk and they are identified by the element type, the element ID and the beat number. The element type determines a certain output list, but several element instances can exist in a working system, therefore, it is needed to know, which instance is in question. Additionally, the beat number specifies the output data for a specific beat.

There are two ways to store the outputs of the sources:
- Sql database: It comes into the picture when the data is relative lightweight and contains only basic primitives (integers, floats or short strings). The Sqlite backend of the QtSql is used, which is a fast and efficient choice for a small database in a standalone file.
- Media files: The image and the sound are frequently used in various systems and they are considered here as heavier binary data. The temporal appearance of the image and sound are video/audio streams and it is straightforward to save them into media files.

The session storage must understand the new data types defined in higher levels, therefore, there is a plug-in interface where the handling of basic types for the Sql database can be written and coder/decoder pairs can be added for the binary types.

It is not granted that outputs are available every beat, hence the relevant records are marked as empty in the Sql database. The coders/decoders for the binary types are implemented fully through the plug-ins and thus the plug-in implementations must pay attention for this issue.

The playback of a session is sensitive for the existence of the sources. These elements must be created with the same type and in the same order to identify the exact matches between the living instances and the saved records. A mismatch between the IDs of the present instances and the IDs in the database causes the termination of the session, because heuristics are not implemented in the session storage to resolve this situation. On the other hand, if some sources do not live, but they have records in the database, those records are ignored and it becomes a good option to suppress particular sources from a playback.

2.7 Debug window

The debug window is a special sink element with GUI to debug the data in the system during run-time, which has been developed after reaching the limits of command line debugging. The GUI is a window and the views of the data models are embedded dialogs showed on the Figure 3. They contain textual or graphical representation of the outputs of the elements and each dialog belongs to one data provided by one element. Dragging and zooming the dialogs make easy to change the subject. On the status bar, the current beat ID and the duration of the previous beat are shown.

Similar to the session storage, the debug window needs to understand the data subclasses coming from higher layers. There is a plug-in interface to register the data subclasses and their views.

In fact, the debugging should have low resource consumption to avoid the influence on the observed system. High CPU load of the frequent dialog
updates has been discovered and led to different strategies to minimize this effect. The dialogs can be zoomed in to inspect their content or zoomed out to the background. In the latter state, the dialog is not updated because the observer does not pay attention to it. The remaining active dialogs are updated maximum in 5 Hz frequency that is suitable for a human observer.

Figure 3: An example for the debug window from the reference implementation. Two dialogs are zoomed in, which belong to two data types named Sensors and RobotState.

3 Reference implementation

3.1 Overview

The control of a robot dog, proposed in this section, is divided into two places. The on-board (URBI) control is responsible for the immediate responses and the reference implementation of the system concept, the AiBO+ (http://aiboplus.sf.net), on the computer takes care of the learning, planning and thinking of the robot. It is a good division between the particular tasks, because the computational and power resources are limited on the robot and constant, however, the computers are evolved year by year and it is easy to switch the underlying technology.

The robot dog runs an URBI server and the computer can connect there via a local wireless network. The sensor data is transferred to the computer and commands are sent back to AIBO. In theory, it would be possible to control the robot over the internet, but the high latencies do not make this solution feasible, because the image, the sound and other sensor data must be transferred to the computer in a reasonable amount of time.

The concept uses a specific term, surroundings, for the external world in the previous sections. From the point of view of the robot, the external environment is sensed with its sensors (camera, infrared, temperature), therefore, the meaning of the “environment” is different for the system concept and the robot, because the concept is one part of the robot brain in the AiBO+.

3.2 State machine

Currently, the AiBO+ implements a basic state machine (Figure 4), which takes into account the daylight/lightning changes. The darkness indicates the night for the robot, it lies down to the ground, the heart beating is slowed down and the motors are switched off. If the sunshine lights the environment of the robot, the beating is accelerated back to the normal speed, the motors of the joints are switched on and the robot stands up.

When the robot stands and the motors are not being moved, a static balance strategy is applied. AIBO can keep the posture against hustle, but its motors are not so fast and strong, therefore, this ability is limited.

Basically, the mood of the robot can be happy or angry. Moving the non-functional joints during sleep or picking up from the ground and carrying frustrates AIBO, the pulse number raises and starts to bark furiously until the disturbance is stopped. The temporal transitions between the states are continuous in place of discrete, since some e-hormones are used in the system, similar to their biological counterparts.

The pineal gland maintains the melatonin level in both the human and the animal brain. When the eyes see brightness, the production is reduced; in the case of the darkness, the production is increased. For the robot dog, the high e-melatonin level corresponds to the sleep mode and the low level to the awaken state. The brightness of AIBO's camera image decreases, the darkness increases the e-melatonin level in every beat and the state of the robot makes transitions between sleeping and awaken states.

Figure 4: The state machine of the reference implementation

The happy-angry states simulate simple artificial emotions. In normal conditions, the robot dog is happy and the relevant e-adrenaline level is low.
However, the disturbance of the robot raises the e-adrenaline level, the heart beating is accelerated. At high level, AIBO barks and shows unhappy face expression with the LEDs on its head. Stopping the disturbance gets back to the normal mood.

3.3 Derived element and data types
The AiBO+ uses the following main data subclasses to represent the sensor data and the states of the robot body:
- **MImage**: Encapsulates one camera image of the robot.
- **MSound**: Encapsulates the microphone records of the robot.
- **MSensors**: The subclass can contain the raw sensor data (leg joints, accelerometer, buttons etc.).
- **MHormones**: The data type of the e-melatonin and e-adrenaline.
- **MRobotState**: It holds the robot state extracted out of the temporal sensor data series (legs state, posture, mental state etc.).

Several element types are defined with specific purposes:
- **MImageSource**: A source, which receives the camera image of the robot and distributes to the other elements.
- **MSensorsSource**: This source element fetches the raw sensor data of the robot, converts to an internal model and sends to the interested elements.
- **MSoundSource**: Fetches the recorded audio from the robot.
- **MInnerWorld**: A normal element, which interprets the sensor data and builds a model of the robot and its environment.
- **MRobotMind**: A normal element to make decisions upon the internal understanding of the world.
- **MBodyControl**: Controls the body of the robot. Only this subclass can send URBI commands to AIBO.
- **MSoundSink**: This sink element plays sound on the speakers attached to the computer.

The AiBO+ also implements the plug-in interfaces for the debug window/session storage and a permanent data storage to save instances of the data subclasses.

4 Conclusion
A new system concept and its reference implementation have been proposed in this paper. The heart beating provides a design principle for the data processing in the systems where the data flow can be standardized and system input can be saved to simulate of the system behaviour again.

The AiBO+ builds a simple system upon the concept and implements a state machine as well as the open-source initiative of the project can encourage other people to develop synchronized systems based on the proposed concept.

Future work can include the extension of the concept to support complicated designs and the reference implementation can also evolve to more complex state machines.

Acknowledgements
Thanks to my family to tolerate my hobby research at home and my company to support some parts of my efforts.

References


Partially separable fitness function and smart genetic operators for area-based image registration – Part 2: new operators

Janne Koljonen
University of Vaasa
P.O. Box 700, FIN-65101, Vaasa, Finland
Janne.Koljonen@uwasa.fi

Abstract

The displacement field for 2D image registration is searched for by a genetic algorithm (GA). The displacement field is constructed with control points and an interpolation kernel. The common global fitness functions based on image intensities are partially separable, i.e., they can be decomposed into local fitness components that are contributed only by subsets of the control points. These local sub fitness functions can be utilized in smart genetic operators. New smart genetic operators are introduced in this paper, which is a follow-up to (Koljonen, 2008) published in STeP 2008 conference. The optimization efficiency of each operator is studied. Moreover, the effect of the spatial frequency of the control points is studied. The results show that the new operators outperform regular crossover and mutation as well as the previous smart operators.

Keywords: computer vision, genetic algorithm, genetic operators, image registration, partially separable fitness function.

1 Introduction

This paper continues the study of partial separability and its utilization in nonrigid image registration by genetic algorithm (GA). In the previous study (Koljonen, 2008), the image registration task and the concept of partial separability were introduced. Hence, the background is only briefly reviewed in this paper. For a more extensive introduction, the reader is referred to (Koljonen, 2010).

The motivation for the previous studies to develop nonrigid image registration methods originated from the need to develop an accurate and robust method to measure the deformation field of materials. One approach to the task is to apply a speckle pattern onto the surface of the material and to monitor it by means of computer vision.

The computer vision task consists of camera calibration (Tsai, 1987) and nonrigid image registration (Hajnal, Hill, and Hawkes, 2001; Zitova and Flusser, 2003). The image registration task, in the context of deformation measurements, has been tackled since Peter’s and Ranson’s study (1982).

Typically, the undeformed image is divided into templates whose cross-correlation with the deformed image is evaluated with different translations and geometrical transformations to search for the maximum correlation. The deformation field can subsequently be computed using the deformation of one template (see, e.g., Vendroux and Knauss, 1998) or the translations of neighboring templates (see, e.g., Su and Anand, 2003).

The accuracy and complexity of the method depend on the template transformation model. Chu, Ranson, Sutton, and Peters (1985) used only scaling, i.e., the model presumes that the deformation was locally constant. Vendroux and Knauss (1998) used a second-order Taylor series. Cheng, Sutton, Schreier, and McNeill (2002) used B-splines to guarantee continuity between the templates.

The Newton-Raphson method has been typically used to search for the maximum correlation (Vendroux and Knauss, 1998; Lu and Cary, 2000). However, if complex transformations are used, the correlation landscape can be multimodal (Koljonen, 2010: 88–91). Hence, the use of global search methods, like genetic algorithms, is well-motivated.

The objective of this study is to improve the efficiency of the GA-based nonrigid image registration. Obviously, the new genetic operators and the concept of partial separability may be applicable in other contexts, too.

This paper is organized as follows: Section 2 reviews the image registration task and the approach to use a genetic algorithm with the partially separable fitness function. New smart genetic operators are introduced in Section 3. Tests and results are re-
2 Image registration, fitness functions, and genetic algorithm

The image registration task and a genetic algorithm (Forrest, 1993) with a partially separable fitness function are briefly defined next. The image registration method consists of the encoding of a nonrigid image transformation, a method for artificial image deformation, a scalar global fitness function, and a search method based on a genetic algorithm. GA in turn consists of a population of trials, a global fitness function, partially separable sub fitness functions, and genetic operators.

2.1 Image registration

Now image registration is regarded as an optimization problem:

$$ T^* = \arg \min_{T_{\text{registration}}} h(T_{\text{registration}}(F_1), F_2), $$

where $h$ is a homology function between two images, $T_{\text{registration}}$ is an image transformation to register images $F_1$ and $F_2$, and $S$ is the search space.

The homology function measures the degree of correspondence between two images. In practice, the homology function is replaced by an objective function that can be computed and that gives estimates of $h$. In this study, the objective function is based on image correlation.

The image transformation $T_{\text{registration}}$ and its search space $S$ should be such that the correspondence between the transformed images, according to eq. (1), can be as close as possible. On the other hand, the complexity of the image transformation model should be as low as possible so that the parameter search can be done efficiently and overfitting to noise is avoided.

In this study, the transformation model consists of a continuous displacement field that is defined by a regular grid of displacement vectors (denoted by $D$) (see fig. 1 a) that are interpolated using a bicubic kernel. Consequently, the image registration method can be regarded as a $3^{rd}$ order method (Koljonen, 2010: 66–69). The displacement vectors are encoded in floating-point numbers.

A genetic algorithm is used to search for the optimal displacement vectors, and thus the optimal $T_{\text{registration}}$. An initial estimate of the displacement field is obtained by a $0^{th}$ order method (Ibid.: 66–69). The other genetic operators are introduced in Section 3.

2.2 Image transformation

Displacements are used to transform the input image $I$ into an artificially deformed image $A_D$, which can be compared with the reference image $R$. The approach that uses inverse transformation (Sonka, Hlavac, and Boyle, 2008; Koljonen, 2010: 10–13) is used, because it is computationally more efficient than the forward method that was used in (Koljonen, 2008):

First, compute the inverse transformation function $(x, y) = T^{-1}(x', y')$ that defines, in subpixel accuracy, were the value of $A_D(x', y')$ is found in the input image $I$. Second, compute $A_D$ at discrete pixel positions. For each position, compute the subpixel intensity at $I(x, y)$, e.g., by bicubic interpolation. Finally, if necessary, quantize $A_D$.

2.3 Scalar fitness functions

The global scalar fitness function is based on the tonal properties of the reference image $R$ and the artificially deformed image $A_D$. With noiseless im-
ages and an optimal solution, $\mathbf{D}_{\text{opt}}$, of the displacement vectors, $\mathbf{A}_b$ and $\mathbf{R}$ should be (almost) identical. In practice, images include noise. Consequently, there is a residual error at each pixel $(x, y)$:

$$\mathbf{A}_b(x, y) - \mathbf{R}(x, y) = \varepsilon,$$  

(2)

Assuming that the noise is independent and normally distributed, i.e., $\varepsilon \sim \text{NID}(0, \sigma)$, a common approach is to minimize the sum of squared differences (SSD) of the images:

$$\arg \min_{\mathbf{D}} \sum_{(x,y) \in \mathbf{A}_b} (\mathbf{A}_b(x, y) - \mathbf{R}(x, y))^2,$$  

(3)

The corresponding global fitness function is:

$$f(\mathbf{D}) = \sum_{i=1}^{N} \sum_{j=1}^{N} (\mathbf{A}_b(x, y) - \mathbf{R}(x, y))^2,$$  

(4)

In order to have a clearer interpretation of the fitness values the root-mean-square (RMS) value of the difference is used to present the values of the global fitness function in the experimental part of this study:

$$g(\mathbf{D}) = \sqrt{\frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\mathbf{A}_b(x, y) - \mathbf{R}(x, y))^2},$$  

(5)

Obviously, minimizing eq. (5) minimizes eq. (4), too. In GA, the global fitness is utilized in the selection operator.

### 2.4 Partially separable fitness function

Due to bi-cubic interpolation, each pixel in $\mathbf{A}_b$ is affected only by the 16 neighboring points of $\mathbf{D}$. This property called partial separability enables to measure the local fitness related to certain input parameters and use it to favor good building blocks in the reproduction phase of the genetic algorithm.

Each control point $\mathbf{d}_{m,n}$ has a local region of influence on the pixels of $\mathbf{A}_b$. Each pixel, to which $\mathbf{d}_{m,n}$ is one of the 16 closest control points, belongs to the local region of influence. However, solving the exact region is impractical.

Therefore, the ideal region is replaced by a square positioned around the exact region is impractical. The horizontal and vertical dimensions $(W, H)$ of the squares equal to the mean horizontal and vertical distances of the translated control points $\mathbf{O} + \mathbf{D}$, respectively. Thus the squares occupy each pixel on average once.

The sub fitness function $f_{m,n}$ related to control point $\mathbf{d}_{m,n}$ is computed as follows:

$$f_{m,n}(\mathbf{D}) = \sum_{y=x_0(m,n)}^{x_0(m,n)+1} \sum_{x=y_0(m,n)}^{y_0(m,n)+1} (\mathbf{A}_b(x, y) - \mathbf{R}(x, y))^2,$$  

(6)

where summing is done over the rectangles shown in fig. 1 b. $a_i(m, n)$ refers to the original $x$-coordinate of a control point and $d_i(m, n)$ to the respective displacement. $W$ and $H$ refer to the mean horizontal and vertical distances of the translated control points, respectively.

### 3 Genetic operators

The sub fitness functions provide a good opportunity to define genetic operators that modify each displacement vector taking into account the local fitness.

At each iteration, one mutation or crossover operation is performed, after which the new trial is inserted to the population. After the insertion, the population is sorted with respect to the global fitness and the trial with the worst fitness is pruned from the population.

Two mutation operators are used. There are also seven crossover operators. Four of them use two parents, while one needs at least 3 and two at least 7 trials. In the selection of parents, it is checked that clones are not used.

In initialization, the $0^{\text{th}}$ order method is used to obtain a good ‘seed’ solution. The seed is mutated by uniform mutation (later denoted by $M_1$) to obtain an initial population with some variation.

#### 3.1 Previous operators

In (Koljonen, 2008), four reproduction operators were used: uniform mutation (denoted by $M_1$) that treats each control point statistically equally and mutates it with a normal distribution $\text{NID}(0, \sigma_{M_1})$; a smart mutation ($M_2$) that applies a larger variance to control points with poorer fitness; uniform crossover ($C_1$) (Syswerda, 1989); and a smart crossover ($C_2$) that utilizes the local fitness estimates to select the best building-blocks from two parents.

$M_2$ was defined as follows:

$$\mathbf{D}_{\text{offspring}}(m,n) = \mathbf{D}_{\text{parent}}(m,n) + f_{m,n} \text{ (parent)} \cdot \varepsilon,$$  

(7)

where $\varepsilon \sim \text{NID}(0, \sigma_{M_1})$.

$C_1$ was in turn defined as follows:

$$\begin{cases} \mathbf{D}_{\text{offspring}}(m,n) = \mathbf{D}_{\text{parent}}(m,n) & \text{if} f_{m,n} < f_{m,n} \text{ (parent}_1), \text{ (parent}_2) \end{cases}$$

$$\mathbf{D}_{\text{offspring}}(m,n) = \mathbf{D}_{\text{parent}}(m,n) \mathbf{D}_{\text{parent}}(m,n) \cdot \varepsilon,$$  

(8)

where $\Delta$ is the (non-normalized) level of indeterminism. Now $\Delta = 0$. Hence, the else case never occurs.

Next five novel crossover operators are introduced.
3.2 Crossover 3: average

This operator (C3) combines two trials by taking the average of the displacement vectors. C3 does not utilize local fitness estimates.

\[ d_x^{offspring}(m,n) = \frac{d_x^{parent1}(m,n) + d_x^{parent2}(m,n)}{2} \]
\[ d_y^{offspring}(m,n) = \frac{d_y^{parent1}(m,n) + d_y^{parent2}(m,n)}{2}. \]  

3.3 Crossover 4: better gene + directional mutation

This operator (C4) is a combination of C2 and directional mutation. For each displacement vector, referred as gene, the best gene, according to the local fitness, is inherited to the offspring like in C2. The offspring is subsequently mutated in an explorative way, i.e., the direction of mutation is away from the inferior gene. The direction is a rough estimate of the gradient direction. Supposing that \( \Delta = 0 \), C4 can be written as:

\[
\text{if } f_{m,n}(\text{parent1}) - f_{m,n}(\text{parent2}) \leq 0, \text{ then }
\begin{align*}
\text{if } d_x^{parent1}(m,n) > d_x^{parent2}(m,n) \text{ then } \\
\quad d_x^{offspring}(m,n) = d_x^{parent1}(m,n) + f_{m,n}(\text{parent1}) \cdot \varepsilon \\
\text{else } \\
\quad d_x^{offspring}(m,n) = d_x^{parent2}(m,n) + f_{m,n}(\text{parent2}) \cdot \varepsilon
\end{align*}
\]
\[
\text{else } \\
\begin{align*}
\text{if } d_y^{parent1}(m,n) > d_y^{parent2}(m,n) \text{ then } \\
\quad d_y^{offspring}(m,n) = d_y^{parent1}(m,n) + f_{m,n}(\text{parent1}) \cdot \varepsilon \\
\text{else } \\
\quad d_y^{offspring}(m,n) = d_y^{parent2}(m,n) + f_{m,n}(\text{parent2}) \cdot \varepsilon
\end{align*}
\]

3.4 Crossover 5: parabola fitting

This operator (C5) uses at least 7 parents. For each displacement vector, i.e., gene, C5 fits a parabola by the least squares method using the sub fitness function estimates of the parents. Subsequently, to coordinates of the minimum of the parabola are located analytically. These coordinates are used to compute the displacement vector of the offspring. Consequently, C5 is evidently a greedy operator that tries to find to local optimum of each displacement vector in one iteration.

### Table 1. Values of some GA parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \sigma_{M1} )</td>
<td>1.0 pixels</td>
</tr>
<tr>
<td>( \sigma_{M2} )</td>
<td>0.033 pixels</td>
</tr>
<tr>
<td>( \sigma_{C4} )</td>
<td>0.01 pixels</td>
</tr>
<tr>
<td>( \sigma_{C6} )</td>
<td>0.01 pixels</td>
</tr>
<tr>
<td>( \sigma_{C7} )</td>
<td>0.01 pixels</td>
</tr>
<tr>
<td>( \Delta )</td>
<td>0</td>
</tr>
</tbody>
</table>

3.5 Crossover 6: gradient

This operator (C6) needs at least three parents. For each gene, C6 computes the normalized gradient vector using least squares fitting and the sub fitness function estimates of the parents. Subsequently, C6 mutates the best gene of the parents to the direction of the gradient as follows:

\[
D_{offspring}(m,n) = D_{best parent}(m,n) + G \cdot \varepsilon, \quad (11)
\]

where \( G \) is the normalized gradient vector and \( \varepsilon \sim \text{NID}(0, \sigma_{C6}). \)

3.6 Crossover 7: best combination

This operator (C7) operates on at least seven parents. It is essentially a combination of C2 and M2. It tries to recombine several trials optimally into one trial, but it adds some noise to the result to avoid premature convergence.

For each gene, C7 selects the best gene from the parents according to the local sub fitness estimates. Subsequently, C7 mutates the best gene of the parents using M2:

\[
D_{offspring}(m,n) = D_{best parent}(m,n) + f_{m,n}(\text{best parent}) \cdot \varepsilon, \quad (12)
\]

where \( \varepsilon \sim \text{NID}(0, \sigma_{C7}). \)

4 Experiments and results

The objectives of the experiments were to test the efficiency of each genetic operator. Hence, the following information needed to be recorded after each fitness evaluation during the test runs: the genetic operator that was used and the resulting improvement to the global fitness with respect to the best fitness of the parents.

Another objective was to study the effect of the spatial frequency of the displacement vector control points.

4.1 Test images

There is no golden standard method to measure the deformation field, nor any benchmark database of test images. Hence, the evaluation of the accuracy and the comparison of results are difficult. One pos-
sibility to evaluate the accuracy is to generate artificial test images, whose deformation field is thus exactly known.

A series of 160 images was created using a seed image and the algorithm proposed in (Koljonen and Alander, 2008). The seed image was taken from a tensile test specimen before any deformation. A random speckle pattern was applied to the specimen by spray-paint.

The image series was created by an algorithm that tries to emulate the deformation and imaging processes. In the deformation process, the decrease of saturation and the increase of brightness are directly proportional to the local engineering strain. Moreover, the effect of nonuniform illumination is taken into account.

In order to model the effects of defocus and vibration, the image is blurred with a Pilbox filter (see Koljonen and Alander, 2008, for details). Now the radius of the filter is 3.5 pixels. Finally, image noise is modeled with independent colored Gaussian noise. Now the standard deviation of the noise is 20.

A significant benefit comes with the use of artificial test images; the homology function is known. Consequently, the accuracy of the fitness function, which is used to estimate the homology function, can be computed.

The objective of the image registration is to determine the correspondence between the seed image, i.e., the reference image $R$ and the last artificially deformed image, i.e., the input image $I$ (fig. 2). Due to the tonal changes in the artificial deformation process, the fitness function (eq. 5) will never reach zero. In fact, the optimal fitness value, corresponding to the homology distance $h = 0$ (see eq. 1), is 10.3 in this study.

The 0th order image registration algorithm uses the intermediate images to determine the seed trial for the genetic algorithm, while the GA uses only the seed image and the last image.

4.2 GA parameters

Table 1 summarizes the values of the parameters of the genetic operators that were used in the test runs. In all test runs, the size of the population was 50 and the total number of iterations was 1000. Moreover, only one genetic operator was applied at each iteration, with an equal probability.

4.3 Spatial frequency

The GA was run using five different values for the spatial frequency of the control points, namely $f_s = \{1/20, 1/30, 1/40, 1/50, 1/60\}$ control points/pixel.

First, the correlation between fitness $g$ (eq. 5) and the homology function $h$ is studied. Because artificial test images were used, $h$ can now be calculated.

A scatter plot of $h$ against the global fitness $g$ is shown in fig. 3. It can be seen that the fitness function gives estimates of $h$ that can well be used to guide the search, although the correlation is partly inadequate. Fig. 4 shows the correlations between $h$ and $g$ with different values of $f_s$. The results give a weak indication that too many control points may result in overfitting the image registration model to specific image noise.

The best trials of each run with different $f_s$ are compared in fig. 5. It can be seen that although the final fitness can be reduced by increasing $f_s$, the homology distance tends to increase, at least after a certain value.
Next three GA runs were performed using always \( fs = 1/40 \) pixels. Fig. 6 shows how the global fitness, diversity of the population, and the homology distance developed during the optimization runs. The diversity of fitness is computed as the difference of the best and worst fitness in the population. This measure is used to estimate the state of convergence.

The development of the fitness and diversity are rather similar from run to run. In contrast, the development of the homology distance has a larger variation. This result indicates that different displacement fields may lead to the same fitness value. It complicates the optimization, because the fitness function is evidently insufficient to estimate the homology function in high precision, which leads to suboptimal image registration. On the other hand, the deviation of the homology distances is approximately only 0.1 pixels, which is satisfactory in many applications.

The deviation of fitness is reduced to almost zero after 400 iterations. There are mainly two possible interpretations:

1) The diversity of the population should be increased, e.g., by applying more mutation during the run or by selecting the parents so that the diversity of the offspring would not be too similar to any trial already present in the population.

2) The optimization run has achieved the minimum of the fitness function. Actually, the fitness value is a little lower than the optimum, although the homology distance is still 0.7 to 0.8 pixels. It is probable that some overfitting has occurred due to image noise. However, this conclusion is only preliminary. In the future, the fitness landscape near the true optimum, where \( h = 0 \), should be examined in order to understand this optimization problem more thoroughly.

### 4.4 Genetic operators

The efficiencies of the seven genetic operators are estimated by calculating the probability that the offspring has a better fitness than any of its parents. This approach is rather partial because a genetic modification may be useful, even though the fitness is not improved immediately.

For instance, the mutation operators hardly ever improved the fitness, but they most probably have an important role in maintaining the diversity of the population, particularly in the beginning of the GA run. On the other hand, some of the crossover operators also included a mutating part.

The three GA runs with \( fs = 1/40 \) were used to compute the probabilities of fitness improvement. First, the probabilities of the runs were averaged to one series of 1000 iterations. Then the series was smoothed by an averaging filter with a window length of 100. The results are shown in fig. 7.

Next the implications of fig. 7 are discussed operator by operator:

- **M₁ (Uniform mutation):** The role of uniform mutation is now mainly in initialization. Its standard deviation was 1 pixel, which is too high for the convergence phase.

- **M₂ (Smart mutation):** This mutation operator did not produce offspring with better fitness values.
However, it may have a role in the maintenance of diversity. In the future, it should be checked whether the offspring obtained by the mutation operators have good enough fitness so that they ‘survive’ in the population. Namely, if they were always pruned immediately, the operator would be useless.

$C_1$ (Uniform crossover): This is the only operator whose usefulness seems to increase towards the end. However, the probability of improvement temporarily drops to zero at iteration 700. The reason for the efficiency of $C_1$ has not been examined.

$C_2$ (Better gene, two parents): This operator is found efficient throughout the optimization runs. Nevertheless, only appr. 40% of the offspring had a better fitness than the best of the parents’, although the operator tries to recombine two parents optimally, with respect to the local sub fitness values. This is probably due to the fact that the fitness function is only partially separable and, furthermore, only an approximation, as discussed in Section 2.4.

$C_3$ (Average genes): The probability to improve fitness is appr. 10% throughout the optimization runs. This operator should also have an effect on the diversity of the population because the genes of the offspring exclusively differ from the parents’. However, averaging tends to drift the control points to the center of gravity of the control points in the whole population.

$C_4$ ($C_2$ + directional mutation): In the beginning of the optimization runs, $C_4$ outperforms $C_2$, i.e., the directional mutation not just affect the diversity but has also an explorative role. Towards the end, the efficiency drops drastically, probably because the standard deviation of the mutation part is too high to the fine-tuning phase.

$C_5$ (Parabola fitting): Parabola fitting is efficient in the beginning, as expected. It utilized the local information of the sub fitness landscapes and creates optimal solution candidates with respect to that. The efficiency of $C_5$ seems to drop the same way as the diversity of the population drops (see fig. 6).

$C_6$ (Gradient + uniform mutation): $C_6$ also utilizes the local sub fitness landscapes in the form of gradient estimates. The iteration step and the energy of the mutation parts strongly affect the success. The probability of fitness improvement starts from 10%, peaks at 20% the same time as the optimization starts to converge, and gradually approaches zero after that.

$C_7$ (Best gene, ≥ 7 parents + uniform mutation): This operator is the best one in the beginning of the optimization runs with a success percent being over 70. After the starts, its success rate starts to decrease, dropping below 10% at iteration 500.

4 Conclusions and future

This paper was a continuation in the development of a GA-based nonrigid image registration method. The method uses control points of displacements, bi-cubic interpolation of both displacements and intensities, intensity based global fitness function, and search of optimal control point positions by a genetic algorithm.

In this image registration task, the global fitness function can be decomposed into local sub fitness functions using the principle of partial separability. The sub fitness functions were utilized in smart crossover and mutation operators.

New crossover operators, most of which utilize the sub fitness functions, were proposed in this paper. The operators can be considered ‘greedy’, as they try to combine optimal offspring candidates by recombining the information of two or several parents. On the other hand, the diversity of the population was enhanced by including a mutation part to some of the greedy crossover operators.

The results show that the new operators have a great probability to produce offspring with a superior fitness with respect to their parents, particularly, at the beginning of an optimization run.

The fitness function needs more attention in the future. Figs. 5 and 6 showed that a better fitness value does not always indicate a better homology val-

Figure 7. Probability of fitness improvement by each genetic operator with respect to time.
ue. Probably due to image noise some overfitting occurs. Furthermore, overfitting is stronger when the degree of freedom of the registration model is increased, as figs. 4 and 5 indicate. In the future, the fitness landscape near the optimum, where \( h = 0 \), should be examined in order to understand this optimization problem more deeply.

Because there are so many tunable GA parameters, it is improbable that their values were even near to the optima. Moreover, a more optimal solution might be to use GA parameters that change during an optimization run, starting from an explorative phase with a higher mutation energy and ending to a fine-tune phase with a lower mutation energy. In addition, different genetic operators are probably more suitable to one phase than the other. This kind of dynamic GA parameter control could be tested in the future.

Acknowledgments

I would like to thank Prof. Juha Röning, Prof. Arto Visala, and Prof. Colin Reeves, the two reviewers and the opponent of my doctoral thesis, respectively, for the valuable comments on my previous research concerning image registration.

References


Online machine vision for elementary engineering courses

Janne Koljonen, Mats Björkqvist, Jarmo T. Alander
University of Vaasa
P.O. Box 700, FIN-65101, Vaasa, Finland
{jako, mbjork, jal}@uwasa.fi

Abstract

Real-time intelligent systems and signal processing are needed in numerous real-world applications. Usually real-time computation requires dedicated micro-processors, integrated circuits, or, increasingly, FPGA circuits to meet the specifications. As for engineering education, it is demanding but beneficial to integrate real-time systems as a part of other topics, instead of teaching them independently. This paper presents a framework for student projects on FPGAs where simple real-time computer vision algorithms are to be implemented and tested. In order to keep the project simple enough and to avoid struggling with the physical interfaces a VHDL module that acquires the input image from a camera and displays the output image in a VGA display is provided to the students. This paper describes the framework and gives distinct subjects for the student projects in online image processing. Furthermore, the pedagogical aspects of the interdisciplinary integration of courses by cooperative group works are discussed, and suggestions how different artificial intelligence related courses could be integrated by student projects are given. Keywords: computer vision, cooperative learning, interdisciplinary education, FPGA, real-time computation, student projects, VHDL.

1 Introduction

The rapid development of technology puts a great challenge for the universities and faculties of technology to educate experts that are capable to bring the development further. In general, an expert of engineering sciences should, at the same time, master the important theory and practice of the field; be prone to search for, evaluate critically and acquire new knowledge independently; be capable to combine knowledge from different sources; apply knowledge into practice; work in teams; be innovative; etc.

The Faculty of Technology at the University of Vaasa is an example how the curricula of universities reflect the requirements of the local industry and, vice versa, how innovation and the adaptation of new technology emerge from the curricula and research of the universities. For over a decade, reconfigurable hardware has been used to some extend by the local industry. As far as we know, however, the FPGA (Field Programmable Gate Array) technology did not establish, particularly amongst the smaller companies, until recently.

An FPGA circuit consists of a regular 2D array of basic reprogrammable logic cells (LC) that are routed by reprogrammable routing channels, and of reprogrammable I/O cells. The structure of the basic unit varies from vendor to vendor (Grout, 2008: 28).

The simple structure of FPGA makes it scalable. The first commercial FPGA was released in 1985 (http://www.xilinx.com/company/history.htm). After that the number of gates that can be implemented in a single chip has increased rapidly. In 1987, an FPGA circuit could implement appr. 9,000 ASIC equivalent gates, whereas the recent Stratix V circuit family from Altera can implement up to 14,000,000 gates (http://www.altera.com/literature/br/br-stratix-v-hardcopy-v.pdf).

Because FPGAs are reprogrammable and yet can implement rather complex functions, they are suitable for versatile prototyping and small-scale production. Examples of FPGA applications are presented, e.g., in a special issue of Microprocessors and Microsystems (2004). Typical applications where FPGAs are used are, a few to mention, high-speed network cards, sound processing cards, smart cameras, and various industrial devices like protection relays.

FPGA circuits are usually programmed using hardware description languages (HDL) like VHDL or Verilog. FPGA circuits are also available, e.g., for LabVIEW thus enabling graphical programming (http://www.ni.com/fpga/).
The growing interest in FPGAs has increased the demand on FPGA experts. One can certainly claim that the companies are currently short of skilled labor in the field. On one hand, in-depth specialization to VHDL (IEEE, 1987) programming as well as managing of FPGA projects are needed. On the other hand, system-level design, broad knowledge of, e.g., signal processing, and bare-minded innovation to apply FPGAs to new applications are awaited. Thus both dedicated courses and interdisciplinary learning should be provided.

In this paper, a problem-based interdisciplinary approach to integrate the teaching of FPGAs and VHDL to other engineering courses is proposed. Integration is done by student projects. Particularly, a project-work framework to a basic course in digital signal processing is introduced. The framework can be used to implement real-time image processing operators.

This paper is organized as follows. Section 2 reviews the literature, and discusses some pedagogical aspects of teaching as well as the interdisciplinary curriculum of digital electronics that is being adopted at the Faculty of Technology at the University of Vaasa. Section 3 introduces a specific student project framework on FPGAs that is used in the basic course in digital electronics and signal processing. Section 4 shortly discusses the benefits and drawbacks of the adopted approach. A summary and future outlines are given in Section 5.

2 Pedagogical aspects

This section discusses the curricular and pedagogical context of the study. The approach to use interdisciplinary student projects is explained and some similar studies in the literature are reviewed. Moreover, suggestions to adopt the approach to the education of various courses related to artificial intelligence are given.

2.1 Related work on interdisciplinary teaching

Previously we have already presented a student project on median filters to integrate digital electronics and signal processing (Koljonen and Alander, 2004). This is a continuation in the attempts to develop problem/project based interdisciplinary learning.

An interdisciplinary and problem-based approach has also been adopted in (González-Vázquez and Loya-Hernández, 2007) and (Loya-Hernández, González-Vázquez, and Garduño-Mota, 2007). Instructors from different fields were also utilized, particularly, in order to promote opportunities to continue the projects on academic research projects or theses. They applied the method to a senior year course that concentrated on reconfigurable system design.

Hall and Anderson (2005) have used an approach similar to ours to integrate DSP theory, VHDL modeling, and hardware synthesis by FPGAs. They have constructed a USB interface from Matlab to FPGA with a VHDL framework on the FPGA, too. Thus the student project is limited to the implementation of a DSP module in VHDL. Moreover, the use of Matlab effectively facilitates the verification of the FPGA design.

2.2 Curriculum of digital electronics

Two courses are dedicated to digital electronics at the University of Vaasa: a basic and an advanced course. The former is mandatory to all engineering students while the latter is an optional course, intended particularly to the students that focus on signal processing and automation.

The basic course deals, e.g., with the basic logic gates, the Boolean algebra, latches, state machines, and the basics of the VHDL modeling language. VHDL teaching focuses on the basic principles and syntax, different levels of abstraction, and simulation of VHDL designs. FPGA synthesis is also demonstrated and tested as a preliminary appetizer. However, the consideration of how to write efficient VHDL for FPGAs falls outside the scope of the course.

When teaching VHDL basics, a great challenge is to divert the students’ way of thinking from sequential programming to the modeling of digital circuits with a modeling language, which is, in some sense, deceivingly similar to the common programming languages, such as Java. Therefore, the digital circuits to be modeled are first designed using pen and paper, after which they are modeled in VHDL, perhaps using several levels of abstraction, and tested by simulation.

The advanced course in digital electronics goes deeper into the VHDL language and its versatile structures. Students should learn to manage a large VHDL project and to design and implement its components generic and reusable when appropriate.

A particular emphasis is put on the understanding of the connection between the VHDL code and the resulting FPGA circuit. The book of Mark Zwolinski (2004) is followed throughout the course when teaching how to model, e.g., efficient synchronous logic without additional latches.

These two dedicated courses give a good foundation to apply FPGAs to applications. Both courses also include a problem-based project work where a larger digital circuit is designed and tested. Typical subjects for the basic course have been, e.g., median filters (Koljonen and Alander, 2004), alarm clocks,
The design and implementation of an application with physical interfacing is usually too broad and demanding a task in a student project that is done as a part of a lecture course. Therefore, simulation is preferred in VHDL model testing.

The FPGA development boards provide a rather convenient way for some interfacing. We use currently DE2 Development and Education Board (Altera), which comes with push buttons, toggle switches, LEDs, 7 segment displays, and clock crystals that are straightforward to use. On the other hand, using, e.g., the LCD display and the coder-decoder (codec) unit is too demanding for the uninitiated, particularly, as a part of a larger design.

Nevertheless, the simple interfaces can also be used for testing of designs, may it be a bit cumbersome. For instance, the frequency meter can be tested by creating periodical signals from the crystal clock signals by frequency dividers that can be controlled by the toggle switches. Moreover, the results can be displayed on the LEDs or the 7 segment displays.

The recent, growing interest in applying FPGAs in companies in different fields has created pressures to revise the curriculum of digital electronics. Instead of introducing new dedicated courses, it was decided, at least to begin with, to expand the teaching of digital electronics to other courses. This is mainly done by adding exercises where, e.g., parallel implementations of digital filters have to be considered and, perhaps, modeled in VHDL, and by using FPGAs in the student projects.

Later the expansion of FPGA education and the integration of different courses by student projects can be taken further. Fig. 1 shows one scenario how to integrate different artificial intelligence related courses by different interdisciplinary student projects.

Because the student projects are carried out applying a problem-based approach, FPGA is by no means the only technology that should be used. Actually, it is beneficial if different hardware is used in different project teams. For example, a median filter for image processing could be implemented in four groups, each using a different technology, namely, FPGA, micro-controller, PC, and a smart camera. When the projects are finally presented, different approaches and technologies can be discussed and compared as for their applicability for different applications.

The basic course in digital signal processing was selected to be the first course where this interdisciplinary approach to teach FPGAs was adopted. It is supposed that students have passed the basic course in digital electronics before attending the course. Thus they should be able to write descriptions of simple digital circuits in VHDL. On the other hand, issues of FPGA synthesis probably have to be taught, at least to some extent, during the guidance process. Alternatively, apt students could just be given references to proper literature.

Traditional programming languages describe what the target device does. With FPGAs and CPLDs (Complex Programmable Logic Device) a hardware description language is used and it describes what kind of a device the target device will be and sometimes also what it does. Due to the great difference between traditional programming and FPGA/CPLD programming it can be really challenging for a beginner to do any meaningful projects using hardware description languages like VHDL.

Figure 1. Examples how to integrate courses by student projects in an interdisciplinary way. The arrows imply the recommended study order of the courses.

<table>
<thead>
<tr>
<th>Course</th>
<th>Topics</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic course in digital electronics</td>
<td>Explore logic gates, Boolean algebra, A/D, D/A, basics of VHDL, Verilog, simulation, etc.</td>
<td>Focus on fundamental concepts and practical implementation.</td>
</tr>
<tr>
<td>Advanced course in digital electronics</td>
<td>Explore advanced, state-of-the-art VHDL, FPGA synthesis, etc.</td>
<td>Extend knowledge for more complex designs.</td>
</tr>
<tr>
<td>Soft computing</td>
<td>Explore introduction to genetic algorithms, artificial neural networks, and fuzzy logic</td>
<td>Introduce non-traditional computing paradigms.</td>
</tr>
<tr>
<td>Fuzzy systems</td>
<td>Explore fuzzy logic, fuzzy control, fuzzy arithmetic, etc.</td>
<td>Develop skills in handling uncertain information.</td>
</tr>
<tr>
<td>Project: Image processing in FPGA</td>
<td>Explore signal processing, image analysis</td>
<td>Apply learned concepts to real-world problems.</td>
</tr>
<tr>
<td>Project: Optimization of a fuzzy system by GA in FPGA</td>
<td>Explore genetic algorithms, differential evolution, ant colony optimization, swarm intelligence, etc.</td>
<td>Combine hardware and software for optimization.</td>
</tr>
<tr>
<td>Project: Genetic programming (GP) in FPGA</td>
<td>Explore genetic programming, advanced topics of nature-inspired computation, etc.</td>
<td>Use evolutionary algorithms for system design.</td>
</tr>
</tbody>
</table>
To aid and motivate the students to apply FPGAs with image processing already in the basic courses, an easy-to-use interface module was created for them to access a camera and a VGA display that were attached to the Altera’s DE2 Development and Education Board. With the interface module the students can implement simple image processing functions that process the image data online.

2.3 Interdisciplinary student projects

If more complicated or self-made interfaces are used or if more complex circuits are designed, a solution is to complete the project as a part of two or more courses, as proposed in (Koljonen and Alander, 2004). Another solution is to provide necessary VHDL components to the students so that the remaining part to be designed and implemented is kept simple enough. The latter approach implicitly teaches the reuse of code.

When an interdisciplinary approach is applied to second year courses, careful attention has to be paid to the design of the projects. They must be simple enough and concentrate on the essential part of the design and implementation issues that are related to the particular courses that are being integrated.

When nearly all automation courses at the University of Vaasa contain a project work, it is reasonable to combine related project works into larger projects shared by both several courses and several students. Therefore it is possible to do a project work covering, e.g., the following courses: Digital signal processing, Soft computing, and Advanced digital electronics.

In the field of artificial, or better computational, intelligence, the following courses are arranged at the University of Vaasa: Soft computing, Artificial neural networks (ANN), Fuzzy systems, Genetic algorithms (GA), and Genetic programming (GP). There is a great potential to provide interdisciplinary student projects to integrate two or more of these topics to digital electronics and real-time computation (see fig. 1).

For example, a project work on genetic algorithms could use GAs to optimize, e.g., some parameters of the designed model, say, fuzzy rules. Digital electronics and FPGAs are naturally integrated to the project work as an implementation platform.

Signal processing is in turn a natural tool for all kinds of measurements as well as sound and image stream handling. One possibility could be to implement neural network based adaptive filters in FPGA (Zhu and Sutton, 2003), thus integrating the courses on ANN, digital signal processing, and digital electronics.

Feed-back control loops and fuzzy control implemented in FPGA could also a topic for a student project. A simple experimental setup would consist of a rangefinder and a step motor. The objective could be to maintain a constant distance to a movable object. Sulaiman et al. (2009) have written a review of fuzzy logic applications and design principles in FPGA. The control loop could also be implemented using fuzzy control rules. The student project would thus integrate, e.g., fuzzy systems, digital electronics, and a course on control engineering.

Martínek and Sekanina (2005) introduced an evolvable image filter that was completely implemented in FPGA. This kind of evolvable hardware (Alander, 2010) could be one topic for an interdisciplinary student project. It could integrate the courses on ANN and GA (or GP) to Advanced digital electronics.

Project works that integrate three or more courses are always tailored according to the personal skills and interests of the student. As a rule this kind of combined project works are done by a single student, not by a group of students. Namely, the larger the project group and the more challenging the problem is, the more likely the group will sooner or later disintegrate.

3 Case: image processing

This section describes how the image processing framework, called DigiPixel, is implemented. Furthermore, some distinct topics for student projects using DigiPixel are given.

3.1 VHDL camera interface

The camera system is built up of three main parts. The hardware consists of:
- Terasic TRDB-D5M camera,
- Altera DE2 development board, and
- VGA display.

Fig. 2 depicts a schematic diagram of the hardware configuration, starting from the left with the camera module TRDB-D5M (Terasic) that represents the ‘eye of the system’, DE2 development board (Altera) that represents the ‘interpreter of the view’, and the VGA display that presents the interpretations.

The camera module includes a CMOS memory that is sensitive to light. The information of CMOS memory can be read out through the camera port. The memory area that represents the image in front of the camera is now handled by the Altera DE2 development board that sends out the processed image through the VGA line to the VGA display to the right in the fig. 2.

Fig. 2 also depicts a block diagram of the important components of the image processing logic im-
implemented on the Altera DE2 development board. The FPGA logic is divided into four design units:

- Image acquisition logic,
- VHDL interface to DigiPixel,
- SRAM, and
- VGA controller.

The image acquisition logic part is connected to the camera via a general purpose parallel input/output port (GPIO) of the Altera DE2 development board. The conversion from the serial form of read red, green and blue pixel values into a more convenient parallel format takes place in the unit, too. This is done by using the control and data bus signals between the development board and the camera. Moreover, the unit needs to take care of some initialization and resetting routines of the camera.

Typically, the implementation of these initialization routines is rather complex when using logical descriptions. Hence, it is not recommended to leave the implementation task to the students when their focus is, e.g., in image processing. Instead, it would be wise to give a pre-implemented framework, as described in this paper.

The VHDL interface to DigiPixel (see fig. 2) can be regarded as a socket where the image processing logic implemented by a student group is attached to. The interface includes a component initialization of an entity called DigiPixel, which defines the ports that are available for the architecture that the students should implement:

```vhdl
entity DigiPixel is
  port(
    Ri : in std_logic_vector(7 downto 0);
    Gi : in std_logic_vector(7 downto 0);
    Bi : in std_logic_vector(7 downto 0);
    Reset : in std_logic;
    Clock : in std_logic;
    Ci : in std_logic_vector(7 downto 0);
    Ro : out std_logic_vector(7 downto 0);
    Go : out std_logic_vector(7 downto 0);
    Bo : out std_logic_vector(7 downto 0);
  );
end DigiPixel;
```

The interface also includes a `for … use` clause that is used to define the architecture that is used in the design:

```vhdl
for ul : digipixel use entity work.digipixel(filter1);
```

Fig. 2 depicts two alternative ways to send the processed pixel values to the VGA controller. The processed pixels can be sent straight to the VGA controller or into a memory, SRAM, from which they can be read by the VGA controller. The use of memory is useful when more complex image processing operations using large neighborhoods are needed.

The VGA controller takes care of the synchronization of several signals that must have an exact
timing in order to obtain a good image in the VGA display. There are mainly signals like hsync, vsync, video_on that are sent all the time to the VGA display. These keep the VGA display ‘awake’. Other signals like red, green and blue pixel value buses are also connected to the VGA output port.

3.2 Subjects for student projects

The architecture of DigiPixel (fig. 2) limits the range of practical image processing operations to: point operations, one-dimensional image filters, and time dependent point or filtering operations. The operations can be controlled using eight signals that are connected to the toggle switches of the DE2 board.

Point operations handle one pixel at a time. They map the original pixel values to the new values by a function that is usually independent of the pixel position (Burger and Burge, 2008: 53–86). Adding time dependency to the mapping function gives more options to the subject of the student projects.

Simple point operations that can be implemented on FPGA are, e.g.: negative transformations, intensity scaling, intensity biasing, threshold operations with different coloring, color replacement by logical conditions, color channel swapping, gamma correction, and color space transformations. In addition, combinations of them can be used.

Spatial filtering operations should be limited to one-dimensional filters, because the pixel data are read in a row-wise matter. 2D filters would require a lot of memory handling, and thus they are not suitable for elementary courses. Nevertheless, even 1D filters can be used effectively to obtain useful real-time image processing with a motivating outcome. A good example is edge detection and coloring based on the threshold operation of the approximate horizontal derivative. In addition, FIR and median filters could be implemented and combined with other image processing operations.

In time dependent operations, the parameters of the image processing operations or the operation itself can change over time. For example, if a toggle switch was on, the threshold could increase by 1 every second. Many kind of blinking effects could also be used.

4 Discussion

Engineering education at the university level confronts a dilemma: the forthcoming experts should ideally have both deep and broad knowledge of various fields of engineering sciences in order to be able to manage the design process of modern technology. For instance, a frequency converter incorporates, among others, power electronics, measurements, signal processing, control, and telecommunication. Moreover, one has to take into account, e.g., mechanical and electro-magnetic compatibility (EMC) aspects of the design.

In the depth-first approach, education concentrates to teach each student one field thoroughly. The necessary breadth of knowledge is subsequently obtained by the cooperation of several experts. However, it would be beneficial if somebody commanded the entirety, at least to some extent.

In the breadth-first approach, students learn something from a sparse set of subjects and, perhaps, some topics in a bit more detail. In our curriculum, both depth and breadth dimensions are handled simultaneously by the student projects that are an integral part of nearly all courses and by the bachelor’s and master’s theses. One obvious benefit of our interdisciplinary ‘learning-by-doing’ approach is the fact that the student projects bind the otherwise apparently dissimilar topics of different courses effectively together while deepening their knowledge in the problem-based task.

Students should also be trained to independently deepen their skills and knowledge on specific areas. According to our experience and Hmelo-Silver (2004), problem-based learning should facilitate flexible knowledge and understanding, effective problem-solving skills, and lifelong learning.

The heterogeneity of the students and, particularly, the differences in their starting skills complicate the design of courses and student projects that motivate all the students. This drawback can be overcome by tailoring the project work topics and objectives for each group. On the other hand, tailoring requires personal guidance by the instructor, who should understand the dynamics of working groups and the principles of cooperative learning (Johnson and Johnson, 1994), and supervise the group accordingly.

Cooperative learning refers to a set of particular forms of teamwork. In traditional teamwork, problems concerning, e.g., free riders frequently occur. In cooperative methods, these can be avoided by ensuring that four conditions, the so called PIES rule, are fulfilled in the group. P = Positive interdependence. I = Individual accountability. E = Equal participation. S = Simultaneous interaction.

It is beneficial to accustom the students to teamwork before the actual student project, which should be completed rather independently. One efficient way is to apply the methods of cooperative learning throughout the course.

In the basic course of digital signal processing, cooperative learning was applied in all exercises. Moreover, one subtopic of the course was completely studied using cooperative group work. The group work consisted of the following steps that together
fulfilled the PIES rule: 1) Independent acquisition of knowledge, 2) preparation of a poster in expert groups, 3) presentation of the posters by a gallery tour where each cooperative group has, ideally, one member from each expert group; hence, each student has to present the poster of his/her group, 4) a written examination of the topics. In fact, there was a written examination every week in the course.

In each phase, the students are rewarded by points that have an effect on the final grade. In cooperative learning, the reward structure has a great role, particularly, if the task in hand is difficult (Vedder and Veendrick, 2003).

Cooperative learning has proved its efficiency in many studies (see, e.g., Slavin, 1995 for a review of the studies). An advantage is that usually nearly all students seem to benefit from cooperative learning. For instance, the most skilled students learn by teaching the others in the group. However, perhaps the most important influence of cooperative learning concerns social and psychological skills. Cooperative learning is based on the topical socio-constructivist learning theory (Hickey, 1997), and, as a curiosity, it could also be regarded as a form of swarm intelligence (Beni and Wang, 1989).

According to the feedback collected from the students the approach and the topics of the student projects are challenging. This can be interpreted both positively and negatively; namely, it is realism that modern technology is challenging. Those who are experts on it must work hard to be real experts. On the other hand, the challenges should not be too demanding.

That is why we have tried to make the courses and their topics as motivating as possible so that the work needed to learn everything also seems to be worth the trouble. A sign of success of this motivation approach was, e.g., that the interest of the students on signal processing increased during the course. This was obviously due to the many practical and interesting applications introduced during the course as well the utilization of cooperative learning.

5 Summary and future

This paper described developments of the curriculum of masters of sciences in engineering, particularly, in automation technology. The objective was to include more implementation of real-time systems to the curriculum without introducing new courses. In particular, real-time machine vision applications are emphasized. This is done by an interdisciplinary approach where courses on different fields are integrated by student projects.

The approach was applied on the basic course in digital electronics where the basics of VHDL are studied, and on the basic course on digital signal processing where the prerequisite is to know only the basics of VHDL. Therefore, an easy-to-use VHDL module for interfacing to the camera and VGA display is provided to the students. Suggestions of image processing subjects that could be implemented using the VHDL module by student projects were given in this paper. Moreover, ideas how the concept of using FPGAs on other courses related to artificial intelligence were given.

While FPGAs are often used in time critical systems, combinatorial optimization tasks form an important potential use of FPGAs in industrial products. To our group the most natural optimization approach is to use evolutionary algorithms, which are well suited for many optimization problems encountered in FPGA development (Alander, Nordman, and Setälä, 1995). Evolutionary methods can also be used on FPGAs as evolvable hardware (Martinek and Sekanina, 2005; Alander, 2010). This optimization direction is one of the future directions in our FPGA education development.

Acknowledgments

We would like to thank Toni Harju for his help in developing the FPGA components, Altera and ABB for providing us the necessary hardware, and the local industrial partners that have declared to support us in the future developments of the FPGA curriculum are acknowledged.

References


Symbolic analysis: From theory to practice

Erkki Laitila SwMaster Oy
erkki.laitila@swmaster.fi

Abstract

All formal science is expressed using symbols. Computer programs are solutions programmed by symbols, too. Hence it is essential to study how far you can get with mere symbols when producing interpretations – without using any other artefacts. The research framework for symbolic analysis has been created using a rather reductionist approach. Based on that framework, a systematic methodology for localizing software problems, named ORT, has been built. That methodology is presented in this paper.

Keywords: program comprehension, source code analysis, core computer science, problem-solving

1. Introduction

The traditional way to analyze real world or a closed world like a typical model is external: one uses a library or a template of a model or a language, which models the objective area from outside – using indirect data structures like XML. Modeling external characteristics and the corresponding behavior is difficult and laborious, because there should be an implicit definition for any kind of dependency and information type in the model. The most challenging thing is to model dynamic features, e.g. objects and their connections, because new dynamic information appears only when using the model.

From the program analysis point-of-view there are many difficulties when expressing semantics for programming languages. Because of that, many libraries and semantic notations have been created: denotational, operational and action semantics as well as attribute grammars and DSM languages. The classical paradigms for dynamic and static analysis are principles to analyze the objective area from outside, too. Hence, they have some common drawbacks like how to get accurate information for a specific request from code or model and how to understand all typical relations between objects including temporary instances.

Symbolic analysis is a novel principle, which tries to meet these challenges by presenting an idea of a virtual model, in which every symbol has a complete identity and semantics as well as an own kind of a finite automaton [6] for executing the symbol. In the JavaMaster-tool [2] the artefact Symbol has been programmed as an atomistic hybrid object, AHO, which is capable of using object reference semantics in connecting symbols with each other. By adding these two principles we have succeeded in creating a universal cognitive virtual model, which is an independent closed world for modeling formal systems.

The virtual closed world produces practical value by means of its internal formalisms and generic mental models, enabled by it. Using that principle it is possible to build tools, which automatically can locate critical parts of software and model the most relevant areas of a user problem. The ORT technology (Observation-Reasoning-Technology) uses these principles [3].

2. Technology spaces of symbolic analysis

In research [1] from 2005 to 2008 we introduced four focus areas in order to establish the methodology for symbolic analysis [1]. Those areas are later called technology spaces (TS), because each of them has presentations, tools, and practical uses typical for it. They cooperate in symbolic program analysis as
Code is read into computer memory using the GrammarWare - TS, which abstracts code symbols using a high level language, called Symbolic. In the modeling space (ModelWare) an atomistic model is weaved from the symbolic notation. In that model all dependencies of code have been transformed to atomic formulas, which is a specific clause-notation in Symbolic. For each atomic formula in the model there is one specific AHO object, which then corresponds to the original symbol in the code and symbolic analysis.

There is a specific technology space, SimulationWare, to enable simulating symbols. The output of simulating corresponds to that of dynamic analysis, although some information (symbols) can be missing from it as many times not all information from the final environment and software installation is known. That is why there should be some techniques to retrieve information from the symbolic model and the external environment made by the user. This area is called KnowledgeWare. For that purpose, e.g. for program comprehension, the symbolic technology proves to be rather useful, because the atomic formulas connect symbols in the logical level in the familiar mathematial notation as in \( A = f(g(Y)) \). In that small example the symbol for \( A \) contains an atomic formula (called command) \( f \), which contains a pointer to the second atom, which has a formula \( g \), which refers to the third symbol, which has the reference \( Y \). It is important to note that this rather short notation, typical for Prolog and predicate logic, is complete to describe the executing chain (trace of symbolic analysis) like a parse tree. It can be calculated and simulated in a similar way to execute mathematical formulas or software with necessary parameters. If we know symbol \( Y \) in the formula and the output for \( A \), we can study and prove the whole chain manually using its redundancy. In the manual interactive proof mode the user is the judge to evaluate correctness of the steps in the chain. In a manual study, understanding side effects is the most critical type of task.

Another example is a short if-statement: \( \text{if } (S1, [S2, S3], []) \). In it \( S1 \) can be any reference into a condition of any size with its child symbols. For example \( S1 \) can be a reference to a variable OldFriend. If it contains a true value, then the execution part \( S2 \) and \( S3 \) is executed. \( S2 \) and \( S3 \) can be for example a reference to System.out.println("Hello").

### 3. From theory to practice

The technology produces interpretations from symbols as follows. In Fig. 1 there is a dataflow from the pragmatic goal to the interpretation step. In many practical situations it is recommended to formulate problems and goals using questions. This will provide answers that are useful in order to get closer to the problem. Queries and answers work as a tranformation media to express symbolic transformation made by the tool. However, these questions and answers are only subsets illustrating some parts of the bigger topic area. That big picture is described next.

#### 3.1 ORT – technology for locating problems

The ORT – technology was created in 2008-2010 as a practical part for symbolic analysis. It contains a classification for problems like bugs and faults with 12 different types (see Fig. 3) and a method framework, which contains 14 method levels for formulating and analyzing each situation (see Fig. 2).

#### 3.2 Formulating a problem

There are five specific method levels for formulating software specific problems: M1 concept analysis, M2 cause-effect-analysis, M3 clustering, M4 slicing and M5 hierarchy. They create a flexible way for the user to get deeper to the problem area, to the code. The last formulating step is a method to confirm that the problem really exists in a specific program slice. That last step can be done using debugger, dynamic analysis or simulation.

#### 3.3 Deep analysis, the iterative interactive process

By confirming that the problem really exists in the selected part, the user is able to ground the problem to a program sequence or a set of symbols. That part is usually called a program slice. The ORT technology provides a set of method stereotypes, captured from traditional program comprehension theories, to support deep analysis [4]: M8 state model, M12 control flow, M13 data flow, and M7 function. The function method has certain features to support analyzing object-oriented behavior and architecture. Furthermore, there is one search-oriented method (M10 side effect analysis) to support holistic understanding in the whole area. Adding holistic and atomistic methods makes a systematic understanding process possible in the ORT technology.
3.4 Virtual architecture and megamodel

In handling and programming models, there is a challenge to define limits and constraints for models. One reason for the small use of UML is the fact that UML diagrams and models are not compatible with each other and another reason is that UML doesn’t support code level modeling. Because of these facts it is not possible for anyone to completely understand code using only UML notation.

To remove those drawbacks, especially the artefact megamodel [5], is a means, which makes it possible to combine different models and to collect new input step-by-step to a cumulative comprehensive megamodel. Because of that principle we have been able to make abstractions for any level of ORT technology (M1..M14) to make them compatible both for the user and the tool to be executed. This abstraction creates a virtual architecture, in which the automation degree can be selected in each case based on current conditions. Using the principles of megamodel the ORT technology becomes tool and language independent.

Because of its computation independent abstraction ORT is a useful principle for any developer and can be used in any architecture – without expensive investments for tools. The starting logic for any problem to classify it into problem categories is shown in Fig. 3. It is based on simple questions for the user.

3 Summary

The Symbolic analysis [1] methodology has succeeded in combining together the most central parts of computer science: grammars, modeling techniques, simulation and executing symbols as well as some primitives for knowledge capture. This compact framework creates an extension and justification for Chomsky hierarchy as follows. Where in the Chomsky model there is a level for each language type and the correspondent automaton, in our extension there is a level for specific types of symbols and a corresponding automaton for them. Because of this higher-granular definition each symbol type has a concrete model to execute it, but a common generic architecture to combine all symbol types. This generic approach makes it possible to unify the principles of human cognition and computer power – symbol by symbol and in a large source code model, too.

The formalism normalizes source code information into an object-based database, which makes it possible...
to build versatile tools using the symbolic atomistic model based on some widely known theories like graph theory, to manipulate semantics of the original code.

There is a specific pragmatic use for the methodology for problem-solving purposes. This is the scope of the ORT technology. Even though the user has no specific tool, it is possible for the user to use all of the method levels of ORT in the practical analyzing process typically used for finding bugs in code. In it the user’s capability of making hypotheses and evaluating them is rather import, and in it the user needs some specific mental models, which are created by the ORT methods.

Acknowledgements
The research and its practical approach have been supported by SwMaster Oy.

References


Solving Rubik’s Cube with Genetic Algorithm

Timo Mantere
University of Vaasa, Dept. of electrical and energy engineering
PO Box 700, FIN-65101 Vaasa
timan@uwasa.fi

Abstract

In this paper we study the solving of Rubik’s cube with genetic algorithms. The Rubik’s cube is scrambled with certain number of random turns and GA then finds the sequence of moves needed in order to solve the puzzle from that situation. Results are analyzed as a function of random scrambling turns; the problem obviously becomes more difficult with more turns.

1 Introduction

This study is follow-up to our ongoing hobby of solving Sudoku’s with evolutionary algorithms (Mantere and Koljonen 2007, 2008, 2009). This time we wanted to experiment with some other combinatorial problem and select the Rubik’s cube, since according the amount of research papers it is much less studied mathematical puzzle problem than Sudoku.

The Rubik’s cube is a mechanical 3-D puzzle invented in 1974 by Ernő Rubik, Hungarian sculptor and professor of architecture. It became huge phenomenon and popular toy in 1980’s. It have been estimated that 350 million cubes have been sold worldwide, which makes it the most sold puzzle game in the world, and it is also widely considered to be the best-selling toy in the world.

Despite of its popularity it is rarely used as example problem with optimization and puzzle solving algorithms. In research database there is only 8 papers that mention it, from which only one uses evolutionary algorithm approach. In comparison there are 38 papers mentioning Sudoku, from which 7 have used evolutionary algorithms.

In a basic classical 3×3×3 Rubik’s cube there are six faces which all possess nine 3×3×1 subcubes, the whole cube consists of 26 subcubes, since the middle cube remain still and forms 3-D axes around which the other subcubes rotates. The middle subcube of each face is also just façade connected to the center piece. The other 20 subcubes are unfastened.

The cube is constructed so that all the six faces possess different color (same for each subface on that face), when the puzzle is scrambled the colors get mixed, and the puzzle is solved when each of the six faces possess only one uniform color.

The cube has total of 4.33×10^19 different states. However, it have been demonstrated that the cube can be solved from any state with 22 moves or less.

The cube is rotated so that one face at the time is rotated 90, 180 or 270 degrees. There are three axes and each possesses three slices (if we consider also the middle slice independent). Therefore, we could select from nine slices to which one to turn and from three rotation angles for each. So each time we have total of 27 different turns to choose from. Sometimes the 180 degree turn is considered as two consecutive 90 degree turns, which would reduce the possible moves to 18.

As an optimization problem Rubik’s cube is solved so that we try to find a sequence of turns that solves the cube from scrambled starting situation.

Figure 1: The Rubik’s cube. Face colors marked as A, B and C, edges (axes) as i, j and k, and slice indices as 0, 1 and 2.
1.1 Genetic algorithms

Genetic algorithms (GA) are a set of optimization methods that derive their inspiration, methodology and procedures from the evolution that happens in the nature. With GAs the problem is expressed as a vector containing parameter values that presents the search space of the problem in hand. The set of possible solution vectors are called population, and GA does crossovers between population vectors in order to generate new trial vectors, also mutations (random changes to some parameter values) are used. Usually GAs are applied for optimization and finding the best set of parameters to the certain problem. The fitness function, e.g. the value of a function we optimize guides the search so that those vectors that achieved best values for fitness function has highest probability to survive in the population and also has the highest probability to be parents for new vectors. Those vectors with worst fitness value are removed from the population, and they also do not have chance to reproduce and spread their genetic material. When we do this kind of reproduction and selection long enough the vectors that are in the population are usually relatively good, sometimes even optimal, solutions for the problem. Genetic algorithms are best suited for hard multiparameter optimization problems to which no exact mathematical solution method exist, or finding the exact optimum takes a way too much time.

1.2 Related work

We found only one paper solving Rubik’s cube with genetic algorithms (Stumpf et al 1994). They tried unidirectional, path from start to goal, and bidirectional, two sequences from start and goal that meets half way, search strategies. Their results showed that bidirectional search was more effective. However, neither was able to solve all tested Rubik Cube situations.

2 The proposed method

The Rubik’s cube solving with combinatorial binary GA was implemented with Java programming language.

2.1 The presentation of cube

The cube was programmed as 5×5×5 array, where inner 3×3×3 were used for rotating the cube, the outer layer was used for marking surface colors. Since the corner pieces can be in three different angles it is not subsequent to present cube with just 3×3×3 array. The outer 5×5×2 was always rotated together with inner 3×3×1 slice rotation. The middle 3×3 of 5×5 present the colors of the rotating face, while the outermost values presented the colors behind edges of the actual cube.

Figure 2: The Rubik’s cube opened. Face colors marked as A to E, edges (axes) as i, j and k, and slice indices as 0, 1 and 2, also the possible rotations of A around k axe are shown.

2.2 The used genetic algorithm

With this problem we used binary coded genetic algorithms where each gene was 6 bits long. One gene represented one turn of the cube. First two bits presented around which axel we turn the cube:

- 00 = i axle
- 01 = j axle
- 10 = k axle
- 11 = none

The middle two bits presented which slice is turned:

- 00 = first slice from the left
- 01 = the middle slice
- 10 = last slice, the rightmost
- 11 = none

Last two bits presented the turn angle:

- 00 = 90 degrees
- 01 = 180 degrees
- 10 = 270 degrees
- 11 = no turn (360 degrees)

From this you can see that this gene representation uses only 27 numbers of 64 possible (N27, R27), the rest of the binary numbers are empty moves which we do not count to the moves performed in order to solve the cube. However, the fact that mutation can always change some move to empty move or empty move to actual move seems to help the algorithm operation.

Each chromosome is constructed from 10 or more genes. Chromosome represents the move sequence needed in order to solve the cube. At the beginning chromosome is ten genes long, but the length of the chromosomes is extended by one after each reinitialization.
We also performed test runs with 9 (N9, R9) and 18 (N18, R18) move alternatives. With these there were only 5 bit long genes, where first 4 bits operates exactly as with 6 bit long representation. With 9 moves the last bit represented 0: 90 degrees turn, 1: no turn, and with 18 moves the last bit means 0: 90 degrees, 1: 270 degrees (-90 degrees).

We used two different ways to solve cube, normal (N9, N18, N27), where scrambled cube was solved, and reverse (R9, R18, R27) where solved cube was turned until scrambled situation was found (opposite order of turns is the solution path).

### 2.2.1 Reinitialization

In the combinatorial optimization there often occurs a situation that the optimization does not advance anymore, i.e. we got stuck to the local optimum and can’t get out of there. There exist different strategies to handle that situation, like adding backtracking and hill-climbing. With evolutionary algorithms most effective way, however, is population reinitialization, in other words start over with a new population. New population can be generated randomly of by using some knowledge of the problem that we have collected from the optimization run so far, “cultural knowledge”, we use that strategy with Sudoku problem (Mantere and Koljonen 2008, 2009), but with Rubik’s cube there is no obvious cultural knowledge we could collect and utilize, so the reinitialization was random.

We tested out several reinitialization rules, out of which we finally selected one where counter starts at the beginning of the optimization run, and after fitness function first time reaches fitness value 5 or less we set the reinitialization mark to 3xcurrent_counter_value. This means that we continue the optimization only three times as long, if solution is not found and after that we reinitialize the population and reset counter and reinitialization mark.

After each reinitialization the chromosome length of population members increases by one. This is done since the need of reinitialization might be an indication that we have too short chromosomes.

We run the optimization till we found solution or maximum of total of 10 million trials, after which we give up and count that Cube position unsolvable for our algorithms. During this max. 10 million trials different amount of reinitializations occur with random intervals due our flexible reinitialization rule.

### 2.2.2 The fitness function

We tested several different fitness functions, i.e. counting how many face is incomplete or counting dominate color of each facelet and the amount of different colors on a facelet etc. and sums of different aspects.

The fitness function we selected and used in test runs presented here compares the color of each location of the facelet to the middle location color, if they were different one penalty points is added to the fitness function. All 6 facelets will be counted and the sum is our fitness function value.

Obviously several chromosomes might have same fitness value, therefore we also have secondary criterion that defines that a sequence needing less actual turns is always better than a sequence with more moves.

### 2.2.3 Genetic algorithm parameters

We tested several population sizes between [10, 400] and different amounts of elitism [1, 50] %. In the results presented in this paper we use population size 20 and elitism 1 (5%), i.e. only the best individual survived to the next generation.

All the new individuals were generated so that we first perform crossover between 2 old individuals and the performed mutations to some of the genes in the new individual. Only one individual is generated by each crossover, we did not take the “opposite” chromosome to the new population.

Mutation rate was 5%, meaning that 5% of the genes in the new chromosome got mutated, and from these half got random mutation, another half was mutated so that one randomly selected bit was flipped.

The better old individuals were favored by selecting the two mating chromosomes with the following code:

```java
for(a=POP_SIZE-1;a>=ELITISM;a--){
    x1=(int)((double)(a)*Math.random());
    x2=(int)((double)(a)*Math.random());
    ...
}
```

The code allows both parents to be the same old individual and with the last new individual both parents are always the best old individual, in that case the new individuals is generated only by mutation from the old individual.

We also used duplicate prevention so that the new individuals are sent max. 5 more times to the mutation procedure if it is still identical with the one of its parents. After 5 tries it is unlikely that new individual is still identical, but if that happens the reinitialization will cure the population later, if it gets too degenerated.

The length of each optimization run was the solution found or max. 10 million trials without solution.
3 The results

The results are presented so that we first scramble the cube with $n$ random rotations and then find the sequence of moves needed in order to solve the cube. Each number of $n$'s ($n \in [0, 20]$) are repeated 100 times with different random moves and the results are presented as statistics out of 100 solves as a function of $n$.

From the range on $n$ you can see that we also tested already solved cube ($n=0$) in order to see how long it takes to the algorithm to solve that, and whether it finds an empty sequence or some other sequence that returns the cube to its original state.

We test if the cube is solved after each turn in the case that it get’s solved with less turns that our chromosome offers. However, we do this test after actual turns, so the already correct cube is not tested before first turn is performed.

Fig. 3: The curve of solution efficiencies with different coding, i.e. how many of the scrambled cubes were solved (out of 100). Black lines represent the solving of scrambled cube, gray presents reverse solving, i.e. finding scrambled state starting from solved cube. Numbers 9, 18, and 27 tells which coding was used (number of possible turns, see 2.2).

Fig. 3. shows how many of the 100 tested scrambled cubes with different numbers of scrambling turns each coding were able to solve. The different codings are explained in section 2.2. All different codings can always solve situations where cube is scrambled by 7 or less turns, some of them also 8 or 9 turns. After that the solving efficiency drops. The most efficient coding was N27 (1523 out of 2100), followed closely by N18 (1520). Coding with 9 turn philosophy was most ineffective (N9, 1426) but on the other hand fastest. 27 move philosophy was faster than 18 moves alternative. Also normal solving order (N) was more efficient than reverse order (R). Therefore thereon only N27 results are presented.

Fig. 4 shows a lot of different characteristics measured from the optimization runs. The secondary axis shows that the average trials and time needed to solve cube increases almost logarithmically with more scramble turns. Time and trials are linked to each other therefore their curves are almost identical. All the cubes with 8 or less scramble turns were always solved, but after that the time and trials needed are affected the fact that all cubes were not solved.

Figure 4: Graph representation of the different characteristics as a function of random scramble turns. Black dotted line shows average trials needed to solve cube, and gray dotted line the average time needed. Black lines show minimum, average and maximum turns needed. Gray line shows how many out of 100 were solved (same as in fig. 3), gray comma line shows how many out of 100 were solved without reinitializations, and black comma line how many reinitializations were needed at maximum.

The amount of scramble turns also affects to the amount of solve turns needed, however we never needed more than 23 solve turns with any cube.

Moreover, fig. 4 shows that easy cubes, 4 scramble turns or less, are almost always solved without reinitializations, and needed only one reinitialization max. More difficult puzzles required a lot of reinitializations, and those with 14 or more scramble turns were hardly ever solved without reinitializations.

The numbers that are not present in the figure showed that the last solution found with all simulation runs needed 4 943 938 trials, and we run each optimization until 10 million trials. Therefore there is no need to run more than 5 million trials in the future, since if solution is not found till then, the genetic algorithm will never find it.

Another thing was that most we performed 158 reinitializations, but at worst case solution was
found after 61 reinitializations. In the future there is no sense to continue optimization if solution is not found with 65 or less reinitializations.

Unfortunately we were unable to solve our results with the other known paper studying Rubik solving with Gas (Stumpf et al 1994) since their test positions were not available.

4 Conclusions and future

The Rubik’s cube was much more difficult to solve with genetic algorithms than Sudoku. Basically they possess similar magnitude range of possible solutions. The reason why cube is so much harder to solve with GA is probably that the cube does not give as much guidance information in order to find path to the solution as Sudoku. Also, Sudoku has pattern of numbers and cube has a sequence moves that are much more difficult to find.

We tried several fitness functions during that study, the common property for fitness functions were that they had two goals; try to get as many facelets, of 54, contain correct color as possible, and the other goal was to solve cube as few moves as possible. The subgoals with different fitness functions were to get as many sides, of 6, complete with uniform color. Our different fitness functions were basically different mixes of these goals with different weight coefficients.

Unfortunately no mix worked that well, the reason seem to be that even when the cube coloring is almost complete, even if only two facelets have wrong color, it may need a long sequence of moves in order to swap these wrong colors. Therefore GA easily traps to the local optimum and cannot continue to the solution.

In our test the solving direction did not affect much to the solving efficiency, Stumpf et al (1994) reported that bidirectional search was more efficient than unidirectional searches, but we did not test that kind of optimization.

In the future we plan to add some kind of belief space that would try to recognize intermediate positions needed in order to solve cube. Another plan is to test if the ant colony/GA hybrid that was highly effective with Sudoku solving problem (Mantere and Koljonen 2009) is more effective, than genetic algorithm, also for solving Rubik cubes.

Acknowledgements

Dr. Janne Koljonen is acknowledged for his contribution to the earlier Sudoku solving experiments.

References


Mantere, Timo and Janne Koljonen. Solving and analyzing Sudokus with cultural algorithms. In 2008 IEEE World Congress on Computational Intelligence (WCCI 2008), 1-6 June, Hong Kong, China, pages 4054-4061, 2008.


An Online Evaluation Platform for Proactive Information Retrieval Task

Li Yao⋆
⋆School of Science and Technology, Aalto University
Espoo, Finland
lyao@cis.hut.fi

Antti Ajanki†
†School of Science and Technology, Aalto University
Espoo, Finland
antti.ajanki@tkk.fi

Abstract

The last decade has seen a great progress on the research and applications of information retrieval. The major improvement has been made to combine traditional keyword-based search with implicit inputs such as eye movements or fixations, mouse clicks and voice commands, thus gradually forming a new branch under the name of proactive information retrieval. This paper focuses on the study of eye movement tracking in the document retrieval by constructing a universal online research platform that merges all the research steps - collecting data, feature selection, model selection and testing into a flexible and extendible cross-platform software system.

1 Introduction

Proactive information retrieval (PIR) is a relatively new research field that incorporates explicit user inputs such as search keywords from keyboard and clicks from a mouse with additional implicit inputs such as eye movement, speech, blood pressure and even facial expression. Buscher and Dengel (2009) shows that personalized information, if available, can be combined into PIR and the retrieval shows high efficiency and accuracy. The most commonly used personalization approach is to include a user’s personal information such as age, nationality, sex, educational background and career. However, those kind of information are always too personal to be safely collected. Thus, latest research has been focused on collecting the observation of eye movement that is directly related to a user’s personal interest or attention. Miller and Agne (2005) and many other research discussed the feasibility of the attention-based information retrieval using eye tracker data.

So far, many research has been devoted to studying the implicit feedbacks as a complement of traditional explicit inputs. As a relatively thorough report, He Zhang and Laaksonen (2008) presents a literature survey conducted to review the current state of the art in research concerning the use of eye movement measurements and other non-conventional and implicit relevance feedback modalities in content-based image and information retrieval. In the work of Viitaniemi and Laaksonen (2008), the author presents the results of a series of experiments where knowledge of the most relevant part of images is given as additional information to a content-based image retrieval system using mouse clicks. Furthermore, Arto Klami and Kaski (2008) presents important results on inferring the relevance of images based on implicit feedback about users’ attention, measured using an eye tracking device.

Specifically in terms of PIR using eye movement, according to Campbell and Maglio (2001), the eye movement data is utilized to two very different types of interfaces: command and non-command. Command-based interfaces use gaze location to directly issue commands to the system while non-command interfaces use gaze information to indirectly tune the system to the user’s needs. Command-based interfaces is most conveniently used to control the system such as clicking by fixation. In the field of information retrieval, the non-command interfaces, combination of traditional application with gaze, are used to adapt the computer to return more accurate information. However, gaze data usually comprises large amount of noise due to the flexibility of human’s eyes. Different person may have very different reading behavior and habit that make the universal modeling a non-trivial task. Hardoon and Pasupa (2010) explores the idea of implicitly incorporating eye movement features in an image ranking task by combining image features together with implicit feedback from users’ eye movements in a tensor ranking Support Vector Machine and shows that it is possible to extract the individual source-specific features. Zakria Hussain and Shawe-Taylor (2010) demonstrates that by
using a greedy Nystrom algorithm on the eye movement features of different users, we can find a suitable low-dimensional feature space for learning the individualized behavior.

Based on the principles briefly discussed above, many applications of PIR have been developed for both practical and academic purposes. One of the earliest and most representative system can be found in Maglio et al. (2000) where an attentive information system called “SUITOR” was developed in IBM. The system monitors a user’s behavior of operating a computer such as web browsing, word processing and provides suggestions and helps regarding to a user’s current activity. The main source of information comes from the tracking of the eye movements. In Laaksonen et al. (1999), a PIR system named PicSOM, is introduced for content-based information browsing and retrieval system based on the Self-Organizing Map (SOM). In Laaksonen (2008) and Viitaniemi and Laaksonen (2008), the authors defined and implemented communication principles and data formats for transferring enriched relevance feedback to the PicSOM content-based image retrieval system used in the PinView\(^1\). The goal of PinView is a proactive personal information navigator that allows retrieval of multimedia - such as still images, text and video - from unannotated databases. The modalities of enriched relevance feedback in PinView include recorded eye movements, pointers and keyboard events and audio including speech. Another delicate interactive PIR system can be found in Lszl Kozma and Kaski (2009) which introduces GaZIR, a gaze-based interface for browsing and searching for images. The system computes on-line predictions of relevance of images based on implicit feedback, and when the user zooms in, the images that are predicted to be the most relevant are brought out.

Technically, all the PIR systems discussed above rely on a collection of features which represent the most essential part of information collected from a mouse, a keyboard, a microphone and an eye tracker. The collection of features, on one hand, differentiate individuals from each other and on the other hand, provide enough information for the computer to model the exact behavior or intention of a specific user. Among various of implicit inputs, eye movement tracking is used in different tasks such as document retrieval, image retrieval, and some subtasks such as reading/skimming detection. The selected set of features is coupled with a specific task and a model.

that takes the selected features and generates performance measurements indicating the acceptability of the given feature-model pair. However, in eye movement tracking, the number of collectable eye movement features can be enormous as a result of exquisite structure of human eyes. Furthermore, the number of different models one can utilize is even larger — logistic regression, support vector machine, artificial neural network and their variations — just to name a few. It remains a confusing question that which feature-model-task combination gives the best performance.

The traditional methods to find the best combination are limited in offline mode containing mainly two stages. In the training stage, a task is firstly defined, then data is collected, features are abstracted from the collected data and models are trained using the features. The theoretical performance of the tuned task-feature-model in this stage can be illustrated by using an isolated test set. This is called theoretical performance for the reason that it is entirely based on the data collected from a group of subjects who only represent a small portion of general population. In the testing stage with the fixed task-feature-model tuple, the practical performance test of a task-feature-model can finally be carried out by involving more subjects in the experiment. The actual practical performance is only available by using separate analysis after the whole experiment is finished, which is why this 2-stage process is named after the word “offline”. One may realize that the whole process is time-consuming and involves large amount of work such as constructing an experiment platform for different tasks to collect training data, building and testing different combination of task-feature-model to get theoretical performance measurement and finally analyzing the practical performance. Furthermore, different researchers usually have their own software platform to perform specific tasks and their own models trained in different platforms such Matlab, R or even Python on different operation systems such as Windows and Linux. Thus, a new universal software platform is needed to simplify the whole research process and to provide flexibility and reliability to PIR research. In this paper, we focus on developing such a system having the following promising features:

- It is a proactive document retrieval system using eye movement data.
- Researchers can define different task-feature-model components in their favorite programming languages (e.g. Python and Matlab) and

\(^1\)PinView is an EU FP7 funded Collaborative Project 216529. For more information, go to the project website: www.pinview.eu
components can be easily plugged into the system.

- It incorporates stage one of training and stage two of practical testing into a unified software system that is connected with the backstage research environment.

- It provides online evaluation feedback in the practical testing stage so that the practical performance of a trained task-feature-model tuple is shown in real time during the testing stage while the subjects are performing the experiment.

- Even though the system is made to evaluate task-feature-model combination in an efficient way in the research, the online evaluation feedback can be used in many other purposes such as giving suggestions to a user or providing guidance.

- The system can be easily extended to proactive image retrieval system using eye movement data.

In what follows, section 2 overviews the system and demonstrates the usage of the system. Section 3 goes deeper into the implementation of the system. Section 4 performs the experiment: Reading/skimming detection in order to show exactly how the system works with different task-feature-model components in the PIR research. Note that this work is not dedicated to any specific research task but the emphasis of use of the system in various tasks and how it simplifies the experiment and data analysis and meanwhile guarantees accurate results.

2 System Overview

2.1 High level architecture of system

Figure 1 shows the top level architecture of the system. The system consists of three separate components located in two different operating systems. The Firefox extension is installed in Firefox browser in the Windows computer in which the eye fixations are collected by Tobbi 1750 eye tracker driver and preprocessed to XML-formatted data. The preprocessed XML fixation data is then relayed to the Apache HTTP server.

The Apache HTTP server is installed on the Linux machine where fixation data from Firefox extension is received via standard Common Gateway Interface (CGI) and further processed to generate series of commands sent to the experimental module via socket client/server communication.

The experimental module is designed to run on any Linux machine inside a normal computer network, which is achieved by setting a socket server inside the experimental module and a socket client inside HTTP server. The commands from HTTP server are designed to be executed on specific experimental platforms, which in the current system, are Matlab and Python modules. Take Matlab as an example. The socket server implemented inside Matlab keeps waiting for the message from socket client. The received messages from the socket client are executable Matlab commands containing fixation data. The Matlab commands call the models implemented in Matlab to analyze the fixation data.

Data travels from Firefox to experimental module and then back to Firefox. It has two parts: evaluation feedbacks showing the response and performance of current machine learning algorithm, and the content of the next document. Due to the intensive real-time data communication, a stable network connection and bandwidth with at least 500KB/s is necessary.

2.2 Functional scenarios

Two basic functional scenarios are “collecting training data” and “testing model”. Firefox browser plugin shows a series of documents by using and collects the information of eye fixation data. The fixation data is then used for a research purpose, for instance, modeling a subject’s behavior. After a model is trained, the system is further used to test the performance of the trained model.

2.2.1 Scenario one: collecting training data

A subject selects an interesting topic in the interface shown in Figure 2. Then a series of random documents on that topic are shown to the subject. The
subject reads given documents one by one as in Figure 3. While reading, the subject’s eye movements are collected by the system. The collected data is then used in the research. During this stage, the experimental module does not load any algorithm to analyze the fixations, so the system gives no evaluation feedbacks.

2.2.2 Scenario two: practical test of model performance

After training a model using data from scenario one and plugging it into the system, it is time to run the practical test with new test subjects. In this scenario, the trained model is loaded into the experimental module as in Figure 4. The system collects the fixation data and evaluates the fixations sequentially by using the preloaded algorithm and gives evaluation feedbacks to the Firefox extension. As in Figure 5, the evaluation result is shown at the top of the document. Another alternative way is to play a music note in accordance to the value of evaluation result.

Figure 4: Preparing stage for the practical test after task-feature-model has been tuned with training data. Selecting a topic equals to selecting a model at the backstage.

Figure 5: Subjects read through the documents with model loaded and performance evaluation returned in real-time. Evaluation is normalized between 0 and 1.

3 System Implementation

3.1 Implementation of Firefox extension

Firefox, developed by Mozilla organization, provides the standard of constructing the extension to Firefox browser. In general, every extension consists of two parts, XPCOM and a user interface. XPCOM, which builds a communication channel between eye tracker and Firefox is coded by C++. The implementation of the user interface requires several techniques including Javascript, XML user interface language (XUL) and HTML. XUL and HTML construct the user interface controls such as buttons and labels. Javascript collects raw fixation via XPCOM, and maps each fixation to a paragraph in a document.
Table 1: Functionality of CGI scripts

<table>
<thead>
<tr>
<th>CGI scripts</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>configure</td>
<td>Load experimental module, load model list, configure tasks, (collecting data or testing model)</td>
</tr>
<tr>
<td>collect</td>
<td>Receive fixations, communicate with Python/Matlab, response feedback to Firefox</td>
</tr>
<tr>
<td>format</td>
<td>Generate training data, format all communications to XML file</td>
</tr>
</tbody>
</table>

and transforms the mapping into standard XML format. The XML-formatted mappings are then delivered to HTTP server via Javascript’s embedded object XMLHttpRequest, which is also known as AJAX.

3.2 Implementation of Common Gateway Interface in Apache HTTP server

The Common Gateway Interface (CGI) is a standard protocol that defines how webserver software can delegate the generation of webpages to a console application. Such applications are known as CGI scripts that in principle can be written in any programming language. Apache HTTP Server provides a container to which the Firefox extension sends data by standard HTTP requests. The CGI scripts wait in the HTTP container for the incoming requests, analyze the requests and compose HTTP responses back to Firefox extension. The system includes three CGI scripts written in C, dealing with 3 different tasks in Table 1. Those three scripts explicitly use GNU Cgicc library to traverse XML requests and compose XML response that is then fed back from HTTP server to Firefox extension.

More specifically, configure corresponds to the configuration phase shown in Figure 2 where it dynamically loads the topics or models into the dropdown lists and notifies the server with a subject’s selection after clicking “config server” button.

collect corresponds to the phase shown in Figure 3 and Figure 5. Evaluation result is shown as “Evaluation Result”. When a subject clicks next document button, the request of a new document is inserted into the XML request so that in the corresponding response, not only the evaluation but also the content of new document are provided to Firefox.

format is responsible to convert all isolated XML requests into one well-formatted XML file so that training data are easily collected for the future use.

3.3 Implementation of experimental module

This part gives details of experimental modules. The whole system consists of two experimental modules, Matlab and Python. Each module further consists of several models each of which is in fact a specific machine learning algorithm handling feature selection and classification. Besides, each module has its own implementation of a socket server.

3.3.1 Implementation by Matlab

Matlab implementation of the experimental module makes use of the power of embedded Java network communication package “java.net.SocketServer”, I/O package “java.io” and other utility packages such as “java.lang.String”.

Java receives fixation requests from CGI and judges the type of command from CGI. The “select_task” containing parameters “model” and “task” is always the first command that is relayed from Java and executed by Matlab. It sets up the variable “task” inside Matlab module and loads the model. After that, if the command is either “handle_fixation” or “next_document”, Matlab searches the .m file with the same name as the command and executes the file. If the command is something else other than the previous three, Java will by default recognize that command as the stop server signal and thus stops the server. Basically, any unrecognized commands will stop the Matlab server. For example, “telnet localhost 6666” in Linux console will do. The Matlab server cannot be stopped inside Matlab command line for some safety reasons of the operating system.

From the other way around, when a task is set to “testing”, “handle_fixation” returns evaluation result (otherwise, ”NoEvaluation”) to Java. Java will send back the result to CGI.

Another practical concern is the change of workspace in Matlab. Since in Matlab experimental module, socket server and models are saved under different directories, the change of workspace needs to be explicitly added into the Matlab scripts. The one-server-multiple-models structure relies on accurate change of workspace.

3.3.2 Implementation by Python

The implementation of Python module uses the same one-server-multiple-models structure as Matlab. The difference is that Python is sufficient to handle both
socket communication and model implementation. The command sets are the same as those used in Matlab. Python socket server can be stopped by using keyboard combination “CTRL+C” in Linux console used to start the Python server.

4 Experiment on the system

The importance of the system is that it can be used as an architecture which may also be enhanced and extended in various research tasks. One example is demonstrated in this section. Other experiments may be designed based on the different goals.

4.1 Reading and skimming detection

The last hundred years has seen a lot of research focusing on the behavior of the human’s eyes when reading. The most important results can be found in K.Rayner (1998): When reading silently the eye shows a very characteristic behavior composed of fixations and saccades. A fixation is a time interval of about 200-250 ms on average during which the eye is steadily gazuing at one point. A saccade is a rapid eye movement from one fixation to the next. The mean left-to-right saccade size during reading is 7-9 letter spaces. This depends on the font size and is relatively invariant concerning the distance between the eyes and the text. Approximately 10-15% of the eye movements during reading are regressions. Reading detection provides an accurate way of defining the level of a user’s interest. For example, reading shows one’s consistent interests of the content while skimming shows the opposite. As in Georg Buscher and van Elst (2008), by monitoring the distance and direction in letter spaces, features such as read forward, skim forward, long skim jump, short regression and unrelated move are abstracted and then used to generate both reading and skimming weights for each fixation. Both reading and skimming weights are then added together as a score to judge whether a user is reading or not. A threshold for reading is predefined. If the score exceeds the threshold, system draws the conclusion that a user is reading. If not, a user is skimming. The similar approach is also adopted in Campbell and Maglio (2001).

The differentiation between the human behavior of reading and skimming shows its importance in proactive information retrieval task. After the discussion of the suitable features, the basic classification algorithms are briefly explained followed by the results.

4.1.1 Dataset and feature abstraction

By following the Figure 2 and Figure 3, the data was collected from 2 subjects who was told before the experiment to look for the words starting with capital letter, which mimicks skimming, or to understand the story of the given short passage, which mimicks reading. The whole data set includes 17 documents aimed for skimming and 33 documents aimed for reading. Although with limited size, the data is enough to show the significant use of our system.

In order to simplify the reading/skimming detection task, only document level features that summarize the fixations across the whole document are taken into consideration. The raw fixation directly collected from Firefox extension is of more than 10 dimensions including docID, fixation sequence, 3 pairs of (x,y) coordinates, fixation duration and other trivial information. From the raw fixation, we define new feature tuple $(N, d_1, d_2, D)$ that is considered most useful and informative to this specific reading/skimming detection task:

- $N$: Number of fixations per document
- $d_1$: Average jumping distance of $x$ coordinates between each pair of successive fixations
- $d_2$: Average jumping distance of $y$ coordinates between each pair of successive fixations
- $D$: Average fixation duration per document

The dataset includes 50 rows and 5 columns. Each row $x_i, x \in \mathbb{R}^n$ represents one document. In our following experiment, $n = 5$ is used for simplicity, however, one can come up with more complex features to better represent a document for different purposes. First 4 columns $(N, d_1, d_2, D)$ show the feature tuple collected from the raw data. The last column shows the classification label $c_i$ in which $c_i = 1$ represents reading and $c_i = -1$ represents skimming.

4.1.2 Logistic regression

Logistic regression in Hilbe (2009) is considered as one of the most basic linear classifiers in machine learning algorithms. It tries to construct a hyperplane that separates the instances of two classes by minimizing the classification error. The logistic function $f(z) = \frac{e^z}{1 + e^z} = \frac{1}{1 + e^{-z}},$ where $z = w \cdot x = w_0 + x_1w_1 + x_2w_2 + \cdots + x_dw_d,$ $w_d$ is coefficients and vector $[x_1, x_2, \ldots, x_d]$ represents one observation, is used to decide which class the given instance belongs to. The best model comes from the minimization of the classification error which is defined.
as $e = \frac{1}{n} \sum_{i=1}^{n}(c_i - \tilde{c}_i)$ where $c_i$ is the observation label and $\tilde{c}_i$ is the classification result equal to $f(z)$ and $n$ is the total number of observations.

### 4.1.3 Soft margin support vector machine

Support vector machine (SVM) is another widely used family of classifiers. Linear SVM constructs the hyperplane that separates the given two classes by maximizing the margin between support vectors. Seeking for a plane to separate two classes of instances becomes infeasible when instances are by nature not linearly separable. Soft margin SVM, detailed in Alpaydin (2004), relaxes the separation to allow misclassification. In Figure 6, by minimizing $\frac{1}{2}\|w\|^2 + C \sum_i \xi_i$ subject to $c_i(w \cdot x - b) \geq 1 - \xi_i$ where $\xi_i$ stores the deviation of each instance $i$ from the margin and $C$ is the complexity trade-off, the optimal separation plan $w \cdot x - b = 0$ is computed out by using all the instances between the plane $w \cdot x - b = 1$ and $w \cdot x - b = -1$.

![Figure 6: Soft margin support vector machine.](image)

### 4.1.4 Classification result

The whole data has 50 feature vectors representing 50 documents either by reading or skimming. Their $d_1$ and $d_2$ are plot in Figure 7. According to the plot, we can generally get the sense that reading and skimming are not perfectly separable by straight line. Since the size of data available is relatively small and in order to minimize the prediction error, the expected prediction error is computed for leave-one-out cross-validation. Different combinations of features are also tried with both logistic regression and SVM and the classification result is shown in Table 2.

![Figure 7: Plot of feature subset $(d_1, d_2)$](image)

<table>
<thead>
<tr>
<th>Feature sets</th>
<th>Expected error rate of LG</th>
<th>Expected error rate of SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>0.328</td>
<td>0.325</td>
</tr>
<tr>
<td>$d_1$</td>
<td>0.195</td>
<td>0.225</td>
</tr>
<tr>
<td>$d_2$</td>
<td>0.124</td>
<td>0.135</td>
</tr>
<tr>
<td>D</td>
<td>0.242</td>
<td>0.235</td>
</tr>
<tr>
<td>N, $d_1$</td>
<td>0.147</td>
<td>0.190</td>
</tr>
<tr>
<td>N,$d_2$</td>
<td>0.142</td>
<td>0.130</td>
</tr>
<tr>
<td>$d_1$, $d_2$</td>
<td>0.125</td>
<td>0.162</td>
</tr>
<tr>
<td>N,$d_1$, $d_2$</td>
<td>0.125</td>
<td>0.095</td>
</tr>
<tr>
<td>N,$d_1$, $d_2$, D</td>
<td>0.035</td>
<td>0.085</td>
</tr>
</tbody>
</table>

### 5 Conclusions

In terms of the system, the current performance is acceptable with at least 3 fixations being handled per second both in Python and Matlab modules. This performance is considered as enough foundation to evaluate the advantage and drawback of any given algorithm in real-time. The evaluation feedback is expressed as the music note of different frequencies regarding to the value of the feedback. However, whether the use of music notes is sensible and informative enough to indicate the performance of the algorithm is still under discussion. The future work could consists of finding more reliable approaches.

In terms of the availability of our system, unfortu-
nately, it is not going to be delivered into the public domain quite soon. But its variations have been internally distributed into the research and construction of the PinView system.

Acknowledgements

We would like to give thanks to all the researchers and developers involved in the PinView project. Their outstanding work provides an essential building block of our system.

References


