BSc (Hons) in Computing Science
Proposals for Student Projects

r.b9a6f68, Thu Mar 23 08:27:21 2017 +0100
## Contents

**Part A  Preapproved Proposals**  

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motion Analysis with High Speed Video</td>
<td>6</td>
</tr>
<tr>
<td><em>Johann A. Briffa</em></td>
<td></td>
</tr>
<tr>
<td>A Tool for Stratigraphy Visualisation in Archaeology</td>
<td>7</td>
</tr>
<tr>
<td><em>Johann A. Briffa</em></td>
<td></td>
</tr>
<tr>
<td>Image-based Rendering for Virtual Reality Applications</td>
<td>9</td>
</tr>
<tr>
<td><em>Carl James Debono, Johann A. Briffa</em></td>
<td></td>
</tr>
<tr>
<td>Tracking of Objects in 3D Video Content</td>
<td>10</td>
</tr>
<tr>
<td><em>Carl James Debono</em></td>
<td></td>
</tr>
<tr>
<td>Multi-hop Data Broadcasting in Vehicular Networks</td>
<td>11</td>
</tr>
<tr>
<td><em>Carl James Debono</em></td>
<td></td>
</tr>
<tr>
<td>Towards Automatic Ahead-of-Time Compiler Generation</td>
<td>12</td>
</tr>
<tr>
<td><em>Joshua Ellul</em></td>
<td></td>
</tr>
<tr>
<td>An Android Dalvik Bytecode Interpreter for 16-bit Architectures</td>
<td>14</td>
</tr>
<tr>
<td><em>Joshua Ellul</em></td>
<td></td>
</tr>
<tr>
<td>JavaScript Simulation of Memory Management in Operating Systems</td>
<td>16</td>
</tr>
<tr>
<td><em>Joshua Ellul, Kevin Vella</em></td>
<td></td>
</tr>
<tr>
<td>Super-resolution of License Plates</td>
<td>18</td>
</tr>
<tr>
<td><em>Reuben Farrugia</em></td>
<td></td>
</tr>
<tr>
<td>Automatic Vehicle Detection from Aerial Imagery</td>
<td>20</td>
</tr>
<tr>
<td><em>Reuben Farrugia, Conrad Attard</em></td>
<td></td>
</tr>
<tr>
<td>Fast Deep Neural Network Training on a GPU</td>
<td>22</td>
</tr>
<tr>
<td><em>Reuben Farrugia, Sandro Spina</em></td>
<td></td>
</tr>
<tr>
<td>Ground motion parameter estimation using Machine Learning</td>
<td>23</td>
</tr>
<tr>
<td><em>Reuben Farrugia, Sebastiano D’Amico</em></td>
<td></td>
</tr>
<tr>
<td>Preliminary investigations on Runtime Enforcement Implementations</td>
<td>24</td>
</tr>
<tr>
<td><em>Adrian Francalanza</em></td>
<td></td>
</tr>
<tr>
<td>Proof Rule based Runtime Verification for LTL</td>
<td>26</td>
</tr>
<tr>
<td><em>Adrian Francalanza</em></td>
<td></td>
</tr>
<tr>
<td>Extracting Static information from Memory Dumps</td>
<td>27</td>
</tr>
<tr>
<td><em>Adrian Francalanza, Neville Grech</em></td>
<td></td>
</tr>
<tr>
<td>Expressing Runtime Verification Properties Using Regular Expression in Larva</td>
<td>29</td>
</tr>
<tr>
<td><em>Gordon Pace, Christian Colombo</em></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
A Runtime Verification Tool for Javascript Programs
   Gordon Pace, Christian Colombo .......................... 31
VHDL Code Generator for Digital Filter Implementation
   Trevor Spiteri ........................................... 33
Reusable Building Blocks for Thread and Event Scheduling
   Kevin Vella, Keith Bugeja, Joshua Ellul ................. 34

Part B   Generic Proposals ................................. 36
Algorithm Implementation on Highly Parallel Architectures
   Johann A. Briffa ........................................ 38
Updates to Distributed Simulator for Communications Systems
   Johann A. Briffa ........................................ 39
Simulating formally specified language behaviour and analysis
   Adrian Francalanza ..................................... 40
Empowering incident responders with on-the-field security hardening
   Mark Vella ............................................... 41
Supervisor Index .......................................... 44
Keyword Index ............................................. 46
Part A

Preapproved Proposals
This part of the booklet contains project proposals that have been preapproved by the Computing Science Board of Studies. Students who wish to pursue their Final Year Projects (FYPs) in a topic proposed herein are required to register their interest using the apposite form\(^1\) available online.

\(^1\)http://www.um.edu.mt/ict/UG/DayDegrees/fyps/FYPCS/Registration
Motion Analysis with High Speed Video

Supervised by: Johann A. Briffa
Keywords: Digital Image Processing, Embedded Systems, Software Engineering

The project trials the use of recently available low cost high speed cameras in a sports application and develop the necessary software to enable that use. Similar systems have existed for some time, though at significantly higher costs (in the order of €10–100k per unit). Recent advances in sensor technology also means that cameras are now able to operate with less light, which has always been a limiting factor. While some open source software is already available, its functionality will not necessarily cover all the requirements for the intended application. Therefore, some programming effort will be required to implement the necessary updates. The chosen camera platform exposes the embedded microcontroller, allowing us to modify how the hardware operates, as needed.

Objectives:

• Consider off-the-shelf software for motion tracking (e.g. [1, 2, 3]) and apply this to target archery with the available high speed camera.

• Suggest improvements, listing upgrade features and performing a preliminary analysis of how these could be implemented.

• Design, implement, and test a selection of features identified above.

Student background/interests:

• Interest in image processing and/or embedded systems.

• Aptitude and willingness to program (the project requires the use of C++; prior OO development experience in another language is suitable).

Bibliography


A Tool for Stratigraphy Visualisation in Archaeology

Supervised by: Johann A. Briffa
Keywords: Software Engineering, Archaeology

In archaeological research, the stratigraphic sequence is a key element in defining the context of finds. This is usually visualised as a Harris matrix [1], a graph where each vertex denotes a context and edges define the sequence of contexts. In addition to the direct sequencing, vertices can also have other relationships with each other, for example a number of vertices may potentially represent the same context (i.e. be equivalent). This formal representation is a prime candidate for the use of software support, however the tools that exist for working with these graphs is either commercial, ageing, or no longer available. In any case, no open-source alternative exists, and given the long-term nature of archaeological research (with digs often taking several decades), it is particularly important to consider the readability of electronic data far into the future.

Objectives: This project will seek to implement an open source solution for working with Harris matrices. This includes:

- Determining functional and other requirements from users (making use of existing contacts within the Department of Classics & Archaeology at the University of Malta and with the Pontifical Biblical Institute, Rome).

- Designing and implementing a cross-platform GUI for creating and manipulating the graphs.

- Implementing formal checks on the graph to report on its validity.

Student background/interests:

- Interest in software development.

- Aptitude and willingness to program (the language and platform to be used in the project is still to be determined; prior OO development experience in any language is suitable).
Bibliography

Image-based Rendering for Virtual Reality Applications

Supervised by: Carl James Debono, Johann A. Briffa
Keywords: Video Processing, Digital Image Processing, Computer Graphics

Nowadays, mobile devices are being equipped with more processing power and graphical processing units allowing them to perform more complex tasks. Virtual Reality is a solution that allows the multiple users to observe a stereoscopic image from their viewpoint. Image-based rendering can be used to generate arbitrary viewpoints on the client device [1]. Rendering these virtual views on resource-limited devices, such as smartphones, is very computationally demanding even for the latest devices and drains battery power fast [2]. In this project, the student will study design and develop a lightweight solution that exploits the smartphone’s GPU to provide fast rendering especially during view changes. The developed algorithm will be tested and evaluated in terms of speed and video quality.

Bibliography


Tracking of Objects in 3D Video Content

Supervised by: Carl James Debono
Keywords: Video Processing, Digital Image Processing, Artificial Intelligence

Tracking of objects is an important task in computer vision. It can be used in various applications ranging from robotic vision, surveillance, and human-computer interaction. [2]. Stereo and multi-view video capture the same scene from different viewpoints. Applying object detection and tracking on each view results in high computational complexity. In this project the student will use the colour and the depth information to detect the object in one view [1] and then use geometric relations between the views to identify the object in the other views. Prediction techniques will also be used to estimate the position of the object in time, reducing the search area and hence computations.

Bibliography


Multi-hop Data Broadcasting in Vehicular Networks

Supervised by: Carl James Debono
Keywords: Data Networks, Graph Theory, Artificial Intelligence

New vehicles are being equipped with on-board computers and soon will carry wireless devices that can transmit and receive data. Moreover, vehicles have a number of sensors ranging from telemetry measuring devices to cameras, which are interfaced to on-board stand-alone computer systems. The addition of wireless transmitting devices will allow communications with other vehicles and with the infrastructure and can help in improving road safety. Broadcasting is ideal for vehicular networks since fast dissemination of critical information is crucial for its effectiveness. However, the reliability of transmission cannot be guaranteed in such an environment. The aim of this project is to develop a multi-hop broadcasting protocol [2, 1] and investigate its performance compared to simple flooding of the network.

Bibliography


Towards Automatic Ahead-of-Time Compiler Generation

Supervised by: Joshua Ellul
Keywords: Operating Systems, Compilers, Systems

Intermediate representations (IR) are useful for code distribution due to their platform independence and smaller size. Upon receiving code it is often beneficial to translate received IR into the underlying native architecture instruction set so as to benefit from execution gains (as opposed to having to interpret the IR). This process is called Ahead-Of-Time (AOT) compilation. Writing an AOT compiler is a non-trivial task [3]. It involves mapping from a source language or instruction set to the underlying target native instruction set.

LLVM [5] is a modular and reusable compiler infrastructure that is source language and target architecture independent. New languages can be supported by writing a frontend for LLVM, and new target platforms can be supported by writing a backend for LLVM. LLVM has gained extensive interest from both academia and industry which has resulted in quite a number of frontends [1][2] and backends [4][6] being made available.

In this project we aim to investigate techniques to facilitate automatic generation of AOT compilers for different architectures by analysing how the different input IR instructions are translated to native instructions for different target architectures using available LLVM backends. Heuristics will then be proposed that given the input IR instructions and the generated native code instructions, will output parts of an AOT compiler for the respective architecture.

Bibliography


An Android Dalvik Bytecode Interpreter for 16-bit Architectures

Supervised by: Joshua Ellul  
Keywords: Operating Systems, Compilers, Systems

Over the past decade the Android Operating System has become ubiquitous within mobile platforms. Prior to the Lollipop Android 5.0 release Android’s execution engine, Dalvik, was based on Just-In-Time (JIT) compilation and interpretation of Dalvik bytecode. Thereafter a new execution engine, the Android Runtime (ART), was introduced which performs Ahead-Of-Time (AOT) compilation of Dalvik bytecode to the underlying native instruction set. Although the Dalvik execution engine has been replaced, Dalvik bytecode is still an integral part of Android and the runtime.

Android is also gaining traction within the Internet-of-Things (IoT), specifically on larger devices. The operating system’s footprint is much larger than what many resource constrained systems typically used in the IoT can support. However, the Dalvik bytecode instruction set and a subset of the Android framework could be used on such resource constrained systems. Dalvik bytecode is similar to Java bytecode except that it is a register-based instruction set as opposed to a stack-based one. Previous work has shown that interpretation [3] and dynamic binary translation [4] of Java bytecode are possible on such resource-constrained devices.

In this project we aim to investigate techniques to efficiently interpret Dalvik Bytecode on resource constrained 16-bit microcontrollers. We will aim to specifically implement an interpreter for MSP430 microcontrollers, however will also focus on facilitating other resource constrained microcontrollers. In doing so we will investigate intermediate representation design [1], garbage collection [2] and memory management [3] techniques.

Bibliography


Operating System (OS) concepts are a foundation of many areas of computer science. Understanding what is under the hood can help even high-level programmers to develop efficient code. A practical approach to teaching operating systems by demonstrating internals is beneficial. However available popular operating system source code is often too complex for the purpose of demonstrating internal concepts.

Instructional operating systems specifically designed for demonstration of OS concepts were first proposed in the early 1980’s [3] of which many were motivated by the development of UNIX [2]. Such operating systems required students to install the OS on a separate partition or VM, or required configuration that posed a barrier to entry. More recently web-based OS simulators were proposed. However, they are either no longer publicly available [1] or require browsers that support non-standard frameworks such as silverlight [4].

In this project a client-side JavaScript Operating System Simulator for Memory Management techniques will be developed which will provide implementations and visualisations of the different memory management techniques commonly taught in an undergraduate course. The framework should allow users to plug-in their own implementation of memory management algorithms.

Bibliography


Super-resolution of License Plates

Supervised by: Reuben Farrugia
Keywords: Digital Image Processing, Image Restoration, Super-Resolution

Videos captured by Closed Circuit Television (CCTV) systems are important in many areas such as crime scene investigation and monitoring offences in public roads [6]. These cameras are normally installed to cover a large field of view where the query license plate may not be sampled densely enough by the camera sensors. The low-resolution of the captured images reduce the effectiveness of CCTV in identifying perpetrators and potential eyewitnesses [5].

The goal of super-resolution (SR) methods is to recover a high resolution image from one or more low resolution input images [4]. Class-specific example-based restoration methods have been successfully employed in face super-resolution [2], iris-super-resolution [1] and face inpainting [3] to mention a few. In this project we will investigate the use of class-specific example-based super-resolution methods for license-plate super-resolution. The developed algorithm will extract the low-quality license plate image and perform class-specific example-based super-resolution to increase the resolution and texture detail in the restored image. Experiments will be conducted on publicly available datasets.

Figure 1: Example of License Plate Super-Resolution.

---

2 License Plate Dataset: http://www.zemris.fer.hr/projects/LicensePlates/
3 MediaLab LPR Dataset: http://www.medialab.ntua.gr/research/LPRdatabase.html
Bibliography


Automatic Vehicle Detection from Aerial Imagery

Supervised by: Reuben Farrugia, Conrad Attard
Keywords: Digital Image Processing, Machine Learning, Object Detection

In recent years, the analysis of aerial images taken from drones has become an important issue [2]. The automatic detection of vehicles has various applications, ranging from security, target tracking, traffic monitoring, and automated parking lot detection [1]. The challenges of vehicle detection in aerial surveillance include camera motions, different target resolutions, and illumination variations caused by light (day, night, sunny, cloudy). Fig. 2 shows an example of vehicle detection from aerial imagery.

Recent works on Convolutional Neural Network (CNN) in object detection and identification show its superior performance over classical approaches [3]. In this project, we will investigate the use of deep learning to perform the detection of vehicles from aerial images. For the evaluation, we will use publicly available datasets, such as the VEDAI dataset ⁴ and images captured using a drone at the University of Malta.

Figure 2: Example of Vehicles detection.

⁴VEDAI Dataset: https://downloads.greyc.fr/vedai/
Bibliography


Fast Deep Neural Network Training on a GPU

**Supervised by:** Reuben Farrugia, Sandro Spina  
**Keywords:** Machine Learning, Deep Learning

Deep Neural Networks (DNN) have attained state-of-the-art performance in several application domains [1, 2]. One of the core components of training a DNN is the Backpropagation algorithm which is computationally very intensive. This algorithm iteratively updates the network parameters based on the error observed at the output. When an input vector is presented to the network, it is propagated forward through the network, layer by layer, until it reaches the output layer. The output of the network is then compared to the desired output using a loss function and an error value is propagated back to the input layer. The Backpropagation algorithm uses the propagated errors to update the parameters in an attempt to minimize the loss function.

The parallelism of the large number of identical neurons naturally maps to GPU architectures, which therefore lend themselves perfectly to significantly accelerate the learning process of DNN. A considerable amount of resources, including GPU-based libraries, exist for training convolutional neural networks (ConvNet). This is not the case for deep belief networks (DBN) which will be used in this project. The aim of this work is to implement a fast Backpropagation algorithm for DBNs using GPUs to speed up training. This project is best suited to students with programming experience (C, C++) and with an interest in machine learning and image processing.

Bibliography


Ground motion parameter estimation using Machine Learning

Supervised by: Reuben Farrugia, Sebastiano D’ Amico
Keywords: Machine Learning, Geoscience

The evaluation of the expected peak ground motion caused by an earthquake is an important problem in earthquake seismology and the prediction of the earthquake ground motion has always been of primary interest for seismologists and structural engineers. It is particularly important for regions where strong-motion data are lacking such as the Maltese islands. Planning and design should be based on available national hazard maps, which, in turn, must be produced after a careful calibration of ground motion predictive relationships for the region. Generally, the quantitative estimate of the ground motion is obtained through the use of the so-called attenuation relationships which allow the estimation of specific ground-motion parameter as a function of magnitude, distance from the source, and frequency [1, 2]. These relationships should be calibrated in the region of interest and are usually obtained by regressing a large number of strong-motion data.

In this study we propose to use support vector regression or artificial neural networks in order to retrieve ground motion parameters. These models will be trained and evaluated on data collected by the department of Geosciences at the University of Malta. The model obtained for can be used for upgrading hazard maps and for engineering designs as well. The results obtained in this study are also useful to implement tools like ShakeMap which use this kind of information to generate a rapid estimate of shaking. ShakeMap is a tool used to portray the extent of potentially damaging shaking following an earthquake largely used in for emergency response, loss estimation, and public information.

Bibliography


Preliminary investigations on Runtime Enforcement Implementations

Supervised by: Adrian Francalanza
Keywords: Runtime Analysis, Concurrency

DetectEr and AdapterEr form part of a tool suite for extending a concurrency system (written in Erlang) with runtime monitoring support. They automatically synthesise and instrument monitors from logical specifications that analyse and adapt the system under scrutiny while it is executing. The tool suite has acted both as prototype tool guiding the study of theories for monitoring semantics [3, 4] as well as vehicle for investigating practical issues relating to aspects such as asynchronounous/synchronous monitoring and monitor modularisation [1, 2].

In this project, the candidate will conduct preliminary investigations for an implementation of a prototype tool that generates enforcement monitors. Whereas detection monitors merely flag behavioural violations/satisfactions, and adaptation monitors attempt to change aspects of a running system after a violation/satisfaction is detected, enforcement monitors attempt to anticipate violations before they actually happen. They are thus capable to enforce the satisfaction of a correctness property. The candidate will start from an existing implementation of the tool suite and investigate extensions for enforcement monitor instumentations and enforcement monitor synthesis. Since the monitors will have their own thread of execution that is separate from that of the system being analaysed, the candidate will be exposed to concurrent programming and issues associated with this paradigm such as race conditions.

Bibliography


Proof Rule based Runtime Verification for LTL

Supervised by: Adrian Francalanza
Keywords: Runtime Verification, Logic

Runtime Verification (RV) [2] is a lightweight technique that attempts to verify the correctness of a system by analysing its behaviour at runtime. In most cases, correctness is specified using some formal program logic which a precise semantics, from which monitors are then synthesised to execute along side and analyse the system under scrutiny.

Linear Temporal Logic [3] is a commonly used logic in the setting of RV because its semantics can be given in terms of execution traces. In [1] the authors define a proof system that is attuned to the restrictions of online monitoring (e.g., partial traces and incremental analysis) with the advantage that property satisfactions/violations can be backed up by a proof derivation. They also outline a procedure for extracting monitors from this proof system. The candidate will be expected to understand this formal system and carry out the resp., monitor synthesis from these proof rules.

Bibliography


Extracting Static information from Memory Dumps

Supervised by: Adrian Francalanza, Neville Grech
Keywords: Runtime Verification, Logic

Frameworks such as Android, YARN or J2EE use complex and highly dynamic software engineering patterns such as dynamic proxies, runtime code generation or external services written in external languages, making applications written in these frameworks extremely hard to analyse statically. At the same time, runtime environments such as the JRE or Android RT provide powerful monitoring capabilities that can be leveraged to extract the runtime state of any such application, a simple example of which is dumping the heap.

What kind of dynamic information can be harvested to reveal important aspects about the behavior of programs? For instance, in Android, can the layout of an application and hence some of its control-flow structure, be recovered, if this is stored in some live object structure referenced by a static field, at runtime? In Jython, can Java classes, generated from Python files and stored in the heap be recovered? For YARN, can the inversion of control be resolved by harvesting control flow paths at runtime?

In this FYP the student is expected to first familiarise himself with runtime environments (e.g. the JVM), enterprise application frameworks and systems programming to develop agents (e.g. using JVMTI) to instrument applications and capture and condense their state, while minimising the performance impact. For example, Java agents are executed before an application is loaded and can instrument the bytecode of an application at runtime or can listen to events happening in the JRE whenever a specific debugging event happens (e.g. a new object is allocated). A working prototype of such a tool will be provided, to start with.

Bibliography


Expressing Runtime Verification Properties Using Regular Expression in Larva

Supervised by: Gordon Pace, Christian Colombo
Keywords: Runtime Verification

Runtime verification [3] provides an approach to monitoring the behaviour of a system without directly changing the code thus separating the concerns of normal system behaviour from that of the verification. Starting from a formal specification and a system, a runtime verification tool automatically instruments checks for the properties into the system, ensuring that any violations are identified at runtime.

Runtime verification tools come with their own native property language and although one can typically translate other languages into such the native one, relating errors back to the original property can be challenging. In this project, we will be using the runtime verification tool Larva [2], which uses an automaton-based formalism DATEs to describe properties, and build different ways of allowing properties to be written using regular expressions. Regular expressions have been used in different ways to express properties: (i) positive specifications which state how the system should behave under normal circumstances e.g. the property \( \text{login} (\text{read} + \text{write})^* \text{logout}\) shows that reading a writing should only occur while logged in; (ii) negative specifications which state what should not happen e.g. the property \( (\text{login} + \text{logout})^* (\text{read} + \text{write})\) states that a read or a write while logged out is wrong behaviour. Although equivalent in terms of expressiveness, some properties may be easier to express in one of the two forms. Although translating regular expressions into DATEs can use standard algorithms, relating unexpected behaviour with the original regular expression specification can be challenging. For instance, receiving two consecutive logins would trigger an error in the first property given above, and would have to be explained in terms of the expression by showing that after a login, only read, write or logout should be allowed. The aim of this project is to explore different ways of achieving this.

The project will be organised into phases as follows:

Research and understanding: The student will be expected to read through the papers in the project’s bibliography and identify other work which might be relevant or helpful to understand the tools, algorithms and data
structures involved.

Case study: An appropriate system will be identified — an open source Java project — for which the student will write a number of properties expressed using regular expressions, and implemented directly using DATEs in Larva.

Regular expression monitoring (I): A tool to translate a regular expression (positive or negative) into DATEs using standard techniques from formal languages [5] and using deterministic finite state automata will be built. The translation should ensure that violations of a property will be explained and logged in terms of the original regular expression rather than the DATE produced.

Regular expression monitoring (II): The translation to deterministic automata may lead to an exponential blowup. One way of circumventing this problem is that of producing non-deterministic automata and keeping track of the set of states the automaton may lie in using DATE variables. This approach will be implemented, including feedback using the original regular expression as in the previous case.

Regular expression monitoring (III): Another approach which has been proposed to monitor regular expressions is that of using derivatives [1, 4]. This approach will also be implemented as in the previous two cases.

Evaluation: The different approaches developed will be evaluated on the basis of (i) efficiency in terms of memory and temporal overheads induced; and (ii) complexity of reverse explanation of identified property violations in terms of the original regular expression.

Bibliography


A Runtime Verification Tool for Javascript Programs

Supervised by: Gordon Pace, Christian Colombo
Keywords: Runtime Verification

Runtime verification [3] provides an approach to monitoring the behaviour of a system without directly changing the code thus separating the concerns of normal system behaviour from that of the verification. Starting from a formal specification and a system, a runtime verification tool automatically instruments checks for the properties into the system, ensuring that any violations are identified at runtime.

Javascript is becoming an increasingly popular technology, and its widespread use to interact with web content results in an interaction of technologies which poses particular challenges for runtime verification.

The aim of this project is to build a runtime verification tool for Javascript programs, extended to enable the monitoring of events consisting not only of Javascript control-flow points (e.g. the moment a function is called, or exits) but also upon changes to the Document Object Model (DOM) with which the Javascript code is interacting. The tool will use a guarded-command language to specify properties in the same style as used by the runtime verification tool pollyRV [1].

The project will be organised into phases as follows:

Research and understanding: The student will be expected to read through the papers in the project’s bibliography and identify other work which might be relevant or helpful to understand the tools, algorithms and data structures involved.

Case study: An appropriate system will be identified — an open source Javascript project — for which the student will write a number of properties expressed using guarded-commands in the style of [1].

Instrumenting Monitors in Javascript: One can instrument Javascript programs by directly accessing and changing the global state of the system, which stores the function definitions and which can be modified. Another approach is that of aspect-oriented programming (AoP) [2] — used to inject code into a system to enable verification without having to change the code of the system directly. There are a number of AoP tools for
Javascript, such as Meld meld\textsuperscript{5} and AspectJS\textsuperscript{6}, which the student is to look into and select an appropriate one for building the runtime verification tool upon. The student will explore one or both of these approaches.

**Runtime verification for Javascript:** The student will build a tool for the monitoring of Javascript programs using standard AoP techniques.

**DOM modification events:** Time permitting, the set of events which the tool can handle will be extended to deal with modification of the DOM structure. Different approaches in which this can be done will be explored, but the tool may assume that access to the DOM will be done via jQuery\textsuperscript{7} or a similar library.

**Evaluation:** The tool will be developed will be evaluated on the basis of efficiency in terms of memory and temporal overheads induced in the case study. Other properties which can be scaled up (e.g. monitoring each instance of a DOM object, thus allowing for scaling up analysis to large documents) will also be analysed to give a better picture of the effectiveness of the approach.

**Bibliography**


\textsuperscript{5}https://github.com/cujojs/meld
\textsuperscript{6}http://www.aspectjs.com/
\textsuperscript{7}https://jquery.com/
VHDL Code Generator for Digital Filter Implementation

**Supervised by:** Trevor Spiteri  
**Keywords:** Signal Processing

Digital filters can be implemented either in software on a processor or on hardware platforms such as FPGAs. After the filter coefficients are obtained using filter design techniques or software tools like MATLAB, an implementation strategy is chosen for the implementation. This project involves writing a software tool in a high-level language such as C, C++ or Python that reads the filter coefficients from a file and generates code for a hardware design in VHDL [1]. This generated code will be tested using simulation tools, and (optionally) by synthesizing and running on a physical FPGA. Different filter implementation techniques can be explored [2].

**Objectives:** Develop a software tool that converts a digital filter specification into code in VHDL, a hardware description language. The design is to be verified by running the generated designs in a simulation environment and testing the digital filters.

**Student background/interests:**
- Interest in digital signal processing.
- Ability to program in a high-level programming language.
- Willingness to use some basic VHDL code.

**Bibliography**


Reusable Building Blocks for Thread and Event Scheduling

Supervised by: Kevin Vella, Keith Bugeja, Joshua Ellul  
Keywords: System Software, Multicores

This project will focus on identifying common building blocks that resurface in several implementations of thread and event schedulers [5, 4, 1, 2, 3], isolating these reusable primitives in a library, and quantifying the impact of this reorganisation on application performance.

While thread and event schedulers tend to be packaged as monolithic general-purpose libraries that are isolated from application-specific code, there are also use cases where application performance benefits significantly from hard-coding task or event scheduling in the application itself. An example of this can be found in parallel scene rendering, where it is common to bind the notion of a task to the specific application. This project will investigate an alternate method of structuring the system where a collection of lower-level scheduling primitives are exposed to the application. Empirical measurement will determine whether this compromise can yield comparable performance to hard-coded, application-specific scheduler implementations while maintaining the ease-of-use and generality of a scheduling library.

Work on this project will be organised in the following phases:

- Implement a compendium of generic and task-specific user-level schedulers for threads and events on single and multicore processors using a selection of locking and lock-free primitives.

- Identify and isolate common building blocks in a library and reimplement the schedulers using this library.

- Measure, compare and analyse the performance impact of using the library relative to the original implementations.

An good knowledge of C is required, and experience with x86 assembly language would be beneficial. Background knowledge on system software, operating systems and concurrent systems is desirable. In the course of the project the student will gain experience on system software, concurrent systems, lock-based and lock-free techniques, performance measurement and analysis.
Bibliography


Part B

Generic Proposals
This part of the booklet contains a series of generic proposals which have **not** been preapproved. Together with their supervisors, students may use any of these proposals as a starting point in crafting a specific submission to the Computing Science Board of Studies for approval. Therefore, students interested in basing their FYPs on one of the generic proposals presented herein should:

1. Discuss the proposal(s) with at least one of the listed academic members of staff.

2. Upon acceptance of supervision by the academic member of staff, complete the apposite acceptance form\(^1\) available online.

3. Await notification of approval by the Computing Science Board of Studies.

\(^1\)http://www.um.edu.mt/ict/UG/DayDegrees/fyps/FYPCS/Registration
Algorithm Implementation on Highly Parallel Architectures

Supervised by: Johann A. Briffa  
Keywords: GPU, GPGPU, Parallel Computing, High Performance Computing

CUDA is an interface for general-purpose programming on Graphical Processing Units (GPUs) from NVIDIA. The architecture of GPUs emphasises massive parallelism of arithmetic units at the expense of control units and memory caching. This allows a very high speed-up for classes of computationally-intensive data-parallel problems, often found in scientific computing. This is an implementation project, with a significant research orientation.

Objectives: Implement a compute-intensive algorithm on GPUs and optimize for speed. Suitable algorithms to be discussed with supervisor; possible examples include algorithms used in high energy physics from an ongoing collaboration with the ALICE experiment at CERN. Examples of prior work by the supervisor can be seen in [2, 1].

Student background/interests:
- Interest in parallel computing.
- Aptitude and willingness to program (the project requires the use of C++ and NVIDIA CUDA; prior OO development experience in another language is suitable).

Bibliography


Updates to Distributed Simulator for Communications Systems

Supervised by: Johann A. Briffa
Keywords: Software Engineering, High Performance Computing

SimCommSys is a multi-platform distributed Monte Carlo simulator for communication systems [1]. The error control coding component implements various kinds of binary and non-binary codes, including turbo, LDPC, repeat-accumulate, and Reed-Solomon. This code base has been in continuous development since 1997, and currently weighs in at over 55,000 physical lines of code, written by Dr Briffa and collaborators. The distributed computing component of this code uses a client/server architecture built on TCP/IP to facilitate running simulations on grid resources; this also works well on local clusters.

Objectives: This project will look to extend the existing code base, continuing our previous work in this area. Various extensions could be looked at, including:

- Writing a cross-platform GUI for the simulator (i.e. writing software to create and edit simulation files in a user-friendly way).
- Writing a back-end / middle-ware for matching resources with simulations.
- Adding a result validation component to confirm simulation reproducibility and facilitate the use of public computing.

Student background/interests:

- Interest in low-level computing issues and parallel computing.
- Aptitude and willingness to program (the project requires the use of C++; prior OO development experience in another language is suitable).

Bibliography

Simulating formally specified language behaviour and analysis

Supervised by: Adrian Francalanza
Keywords: Semantics, Process Calculi, Programming Language Design, Concurrency

It is now standard for language designers to formally specify the runtime semantics of a programming language using operational semantics and the static semantics using a type system [2]. It is however a laborious task to get this semantics right and establish properties about it e.g., subject reduction or progress.

PLT Redex [1] is a domain-specific language designed for specifying and debugging operational semantics. From standard reduction rules and (type) derivation rules, PLT Redex allows the designer to interactively explore terms and to use randomized test generation to attempt to falsify properties of the semantics. This allows the language designer to tighten the feedback loop and vet out obvious mistakes cheaply through automated tests.

The candidate will be asked to understand an existing language whose (static and dynamic) semantics is already formally defined. She will then be asked to implement this specification in PLT Redex and analyse the behaviour of the language using the model implemented.

Bibliography


Empowering incident responders with on-the-field security hardening

Supervised by: Mark Vella
Keywords: Cyber Security, Digital Forensics, Incident Response

Cyber security threats are nowadays affecting various aspects of critical business processes and personal privacy/safety alike. The increasing trend in having everything connected to the Internet: from business informations systems to industrial automation/control systems; from personal cloud storage and messaging applications; to banking and shopping accounts; is straining state-of-the-art information security mechanisms. The mushrooming of Computer Security Incident Response Teams (CSIRT) is a direct result of this situation.

In the case of digital investigations not involving cyber-attacks, disk forensics is the holy grail. On the contrary, cyber-attacks nowadays make use of exploits and malware that possibly never touch the disk, or do so but still manage to erase every trail of evidence. This is where the importance of memory forensics becomes paramount. This approach follows the adage: "malware can hide but it must run", meaning that memory forensics revolves around investigating unavoidable artefacts produced by successful security breaches. Once malware is hunted down in memory, say identification of a backdoor residing inside the process of a legitimate process, it must be forensically followed up in order to conduct successful recovery and attribution. This task comprises the use of malware sandboxes complemented with static/dynamic analysis of compiled code in order to reverse engineer its behaviour.

This group of projects is aimed specifically for final year undergraduates to join the CyberForensics research project, with the possibility of following up their work at a postgraduate level. Specifically, these projects address the limited reconfigurability of access control mechanisms that is typically bundled with applications. Exploring solutions for this problem would empower incident responders with on-the-field security hardening options. The overall aim is that of providing access to application memory artefacts through program introspection interfaces across multiple platforms, and eventually also abstracted to the level of a program security policy. A number of studies, each within the context of one specific platform, are to be conducted in order to explore and compare ways for:

- Developing source code-level program introspection patches and exposure

\(^{2}\text{http://staff.um.edu.mt/mvel3/research.html}\)
through an Application Programmer Interface (API).

- Developing binary-level program introspection patches, both on-disk and run-time.
- Abstracting API usage to the level of a security policy language.
- Evaluating the proposed solutions within generic live forensics scenarios as well as within specific cyber-attack scenarios.

The platforms of interest are Windows [12], Linux [10], Android [5, 6, 7] and OpenWRT\(^3\) [9], with the relevant underlying architectures being x86, x86-64, ARM [2] and MIPS\(^4\) [3]. The capability of carving data objects from within the memory of executing processes [8] and performing code patching directly upon compiled code\(^5\), constitute the primary targets of this exploration. Once work on a specific platform has sufficiently matured in terms of infrastructure development, the development of access control policy [1, 4] language recognizers [11] would help incident responders to be more productive when hardening applications on-the-field.

Bibliography


\(^3\)http://openwrt.org

\(^4\)http://www.secforce.com/blog/2014/04/reverse-engineer-router-firmware-part-1/

\(^5\)http://rada.re/r/


Supervisor Index

Attard, Conrad, 20

Briffa, Johann A., 6, 7, 9, 38, 39
Bugeja, Keith, 34

Colombo, Christian, 29, 31

D’Amico, Sebastiano, 23
Debono, Carl James, 9–11

Ellul, Joshua, 12, 14, 16, 34

Farrugia, Reuben, 18, 20, 22, 23
Francalanza, Adrian, 24, 26, 27, 40

Grech, Neville, 27

Pace, Gordon, 29, 31

Spina, Sandro, 22
Spiteri, Trevor, 33

Vella, Kevin, 16, 34
Vella, Mark, 41
Keyword Index

Archaeology, 7
Artificial Intelligence, 10, 11
Compilers, 12, 14
Computer Graphics, 9
Concurrency, 24, 40
Cyber Security, 41
Data Networks, 11
Deep Learning, 22
Digital Forensics, 41
Digital Image Processing, 6, 9, 10, 18, 20
Embedded Systems, 6
Geoscience, 23
GPGPU, 38
GPU, 38
Graph Theory, 11
High Performance Computing, 38, 39
Image Restoration, 18
Incident Response, 41
Logic, 26, 27
Machine Learning, 20, 22, 23
Multicores, 34
Object Detection, 20
Operating Systems, 12, 14, 16
Parallel Computing, 38
Process Calculi, 40
Programming Language Design, 40
Runtime Analysis, 24
Runtime Verification, 26, 27, 29, 31
Semantics, 40
Signal Processing, 33
Software Engineering, 6, 7, 39
Super-Resolution, 18

System Software, 34
Systems, 12, 14, 16
Video Processing, 9, 10

47