BSc(Hons) Computing Science
Proposals for Student Projects

2019/2020
Contents

**Project Proposals** .................................................. 3
Memory Forensics for Cyber Incident Response
  *Mark Vella* .......................................................... 4
ProcessJ to JavaScript translation
  *Kevin Vella* .......................................................... 6
Computation mobility with Javascript
  *Kevin Vella* .......................................................... 8
Occam to Go source-level translation
  *Kevin Vella* .......................................................... 9
Never Ending Learning - In Pattern recognition and Machine learning
  *Adrian Muscat* .......................................................... 10
Web-based Testbed for Visual Relation Detection
  *Adrian Muscat, Chris Porter* .......................................... 12
Image-based Rendering for Virtual Reality Applications
  *Carl James Debono, Johann Briffa* .................................... 13
Object Detection in Light Field Images
  *Carl James Debono* ...................................................... 14
Detection and Classification of Brain Haemorrhage
  *Carl James Debono* ...................................................... 15
Indoor localisation system for the CERN particle accelerator injector complex
  *Gianluca Valentino* ................................................... 16

**Project Areas** .......................................................... 17
Area: Computer Graphics
  *Keith Bugeja, Sandro Spina* .......................................... 18
Area: Runtime verification of smart contracts for blockchains and Distributed Ledger Technologies
  *Gordon Pace, Joshua Ellul* ........................................... 21
Area: Static analysis of Smart Contracts using KeY
  *Gordon Pace, Joshua Ellul* ........................................... 22
Area: The Run-time Performance of Concurrent Programs
  *Kevin Vella* ............................................................ 23
Area: Coding Theory, Image Processing and GPU Computing
  *Johann Briffa* .......................................................... 25
Project Proposals

The following are specific project proposals that have already been approved by the Board of Studies.
Cyber attacks nowadays employ highly sophisticated software artefacts [7, 5] that can persist for long periods of time while evading detection. This is where the importance of memory forensics [10] becomes paramount. This approach follows the adage: "malware can hide but it must run", meaning that memory forensics revolves around investigating unavoidable in-memory artefacts produced by successful intrusions.

Malware sandboxes [3, 6] perform automated malware analysis to generate a number of indicators of compromise and help configure memory scanners to detect ongoing infections. However, a deeper level of analysis that directly inspects malware binary code can lead to the development of advanced memory forensics tools that may even reveal the exact manner how the malicious artefacts are being utilized by threat actors. Such tools can provide comprehensive recovery and possibly even lead to successful attribution during digital investigation by incident responders [11].

This project area comprises the use of binary (or bytecode) analysis/instrumentation techniques [8, 9] and frameworks [4, 1, 2] in order to build memory forensics tools for incident responders. The idea is to leverage the fact that several of the aforementioned malicious artefacts tend to be widely used across attack campaigns, whether for global or targeted ones, while morphing sufficiently to elude rudimentary host and network-level detectors.

Bibliography


ProcessJ to JavaScript translation

Supervised by: Kevin Vella  
Keywords: Compilers, Concurrency  
Level: Undergraduate final year project

Java’s concurrency model is loosely based on Hoare’s monitors [1], which are broadly used despite arguably introducing unnecessary complexity in the design of concurrent systems. The typical scenario entails Java threads migrating between objects each time a method in another object is called, resulting in execution flow resembling a spaghetti of threads crossing objects in a highly unstructured manner [5]. Several alternative concurrency mechanisms have been proposed for Java, most of which are library-based [6, 2].

ProcessJ [4] is an experimental concurrent language designed at the University of Nevada at Las Vegas, loosely characterised as Java with its object-oriented features supplanted by process-oriented ones. Still in its early stages of development, ProcessJ presently targets the Java Virtual Machine. Considering that use in concurrency education is the primary intended use for ProcessJ, it is desirable to also target the Javascript interpreter in web browsers, so as to enable students to write and interact with ProcessJ examples directly in the browser.

A major component of this proposed solution has already been built at the University of Malta [3], a Javascript run-time system that cooperatively schedules concurrent processes on browsers and node.js servers and permits channel communication over web socket connections, between web workers, etc. This project covers the next major step, which is the design and implementation of a transpiler that takes ProcessJ as input and produces ‘multi-process’ Javascript code with JS-CSP as the run-time system and scheduler.

Bibliography


Computation mobility with Javascript

Supervised by: Kevin Vella
Keywords: Web Computing, Concurrency
Level: Undergraduate final year project

Javascript [2], as an interpreted language, permits an executing program to dynamically change its own behaviour by modifying its own source code in-memory. This provides an intriguing opportunity to explore the gradual migration of a running program to a new location and transparent resumption of its computation, without providing said locations access to the program code in advance [1].

This project involves the design and implementation of a Javascript framework that allows developers to migrate the state of ongoing computation together with code across locations, for instance from a node.js instance to a web browser and back. Sample applications that take advantage and demonstrate this feature will be developed. Finally, the performance and scalability of computation mobility in Javascript will be characterised through experimentation.

Bibliography


Occam to Go source-level translation

Supervised by: Kevin Vella
Keywords: Compilers, Concurrency
Level: Undergraduate final year project

In few application domains where safety is highly critical, the programming language Occam [3] is still in use. Occam is however, not easily ported to new platforms, nor is the compiler well maintained. From many aspects, an obvious alternative to Occam is Go [2], but a large code base in Occam would have to be translated into Go, which for safety critical code is non-trivial. The proposed project is to create a transpiler that translates existing Occam code into human-readable Go code.

A similar project is being proposed to computing students at the University of Copenhagen [1]. This commonality is intended to promote opportunities for joint work between interested students at the two institutions.

Bibliography


Never Ending Learning - In Pattern recognition and Machine learning

Supervised by: Adrian Muscat
Level: Undergraduate final year project

Typically in a supervised machine learning setup the model is trained using a fully labelled dataset prior to deployment. If the nature of the data changes, the model is re-trained from scratch or fine-tuned, again with the full updated labelled dataset. This setup has two major problems (a) re-training the model is time-consuming, and (b) collating a labelled dataset is very expensive. Problem (a) is usually attributed to the inherent structure of the model, while problem (b) is related to the learning algorithm. These two problems are usually independent of each other. However when, for example, the output labels in the dataset change the two problems are then related. In this project we will draw inspiration from the fact that humans learn incrementally and not always from labelled data. The term “Never Ending Learning” (NEL) has been coined in an attempt to simulate the human learning paradigm (see example applied in a language setting, [5], and in image setting, [3, 4]). Additionally, humans may even discard labelled data, as incorrect. The general aim of this project is to study and develop techniques, methods and algorithms that mimic or simulate this human learning task. We can ground this problem in the prediction of spatial relations, since this is a multi-label problem (highly suitable for this project) and we also have considerable experience and data in this area, [6, 2, 1]. A good start in this project is to review the literature in semi-supervised learning and life-long machine learning. We finally compare results to the supervised models and conclude on the benefits or otherwise of the NEL model. More concretely the student will develop a NEL model for the spatial relation detection task. This project requires a significant degree of experimentation and benefits from an inquisitive and creative mind.

Bibliography


Web-based Testbed for Visual Relation Detection

Supervised by: Adrian Muscat, Chris Porter
Level: Undergraduate final year project

This project is in the area of machine learning models that predict the relationship in between two objects in an image. For example man wearing hat and person working at the desk. In the former wearing describes the relation between man and hat, while working at describes the relation in between person and desk in the latter. Predicting the relation is a sub-task in the area of image understanding, and finds useful application in visual question answering, image description generation and robotics. Our participation in this area is ongoing (see for example [4, 2, 1]) and we are looking into developing an interactive website that can drive this work forward.

In this project the student will build a web-based test-bed for the development and testing of various Visual Relation Detection (VRD) models, [3]. The test-bed would allow humans to, for example, (a) upload images, (b) annotate an image, (c) rate the labels predicted by the machine models or annotated by humans, and (d) carry out statistical analysis, including inter-rater agreements.

This project involves considerable software development and in particular the integration of ready-made models, including R-CNN or YOLO models and various VRD models. The test-bed will be based on a modular framework, such that models can be inserted and selected as desired, ideally via an API.

Bibliography


Image-based Rendering for Virtual Reality Applications

**Supervised by:** Carl James Debono, Johann Briffa  
**Keywords:** Video Processing, Digital Image Processing, Computer Graphics  
**Level:** Undergraduate final year project

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 images and videos 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**


Object Detection in Light Field Images

Supervised by: Carl James Debono
Keywords: Digital Image Processing, Artificial Intelligence
Level: Undergraduate final year project

Light field imaging is a new technology that allows the capture of more visual information compared to traditional cameras. This captured data allows for improved capabilities for scene understanding and improved performance of computer vision solutions. However, it brings about also challenges for data compression, content processing, and display [1]. Object detection is an important task in many computer vision applications. In this project the student will study, design and develop an object detection solution that exploits the additional information provided by this new image capture technology. The developed solution needs to take into consideration the speed of processing.

Bibliography

Detection and Classification of Brain Haemorrhage

Supervised by: Carl James Debono  
Keywords: Computer Vision, Artificial Intelligence  
Level: Undergraduate final year project

The brain is a complex organ that controls most of the activities of the human body. Damage within the brain can thus have a severe impact on the wellbeing of a person. Haemorrhage in the brain occurs because of blood escaping from the circulatory system and can manifest itself internally and/or externally. Detecting the correct location of a brain haemorrhage and its type is crucial in saving lives and preventing further damage. Computed Tomography (CT) scans provide imagery that can be used for diagnostics to evaluate whether a patient needs surgery or not. Deep learning solutions, such as [2], [1], will be used in this project to detect and classify brain haemorrhages. The CT scans will be obtained from the General Hospital.

Bibliography


Indoor localisation system for the CERN particle accelerator injector complex

Supervised by: Gianluca Valentino  
Keywords: indoor localisation, 3G/4G, radio propagation  
Level: Undergraduate final year project

The CERN particle accelerator complex [1] is composed by many underground caverns, tunnels and rooms. Often, it is difficult to navigate and find equipment and electronics in a timely manner. Access time needs to be minimized due to the presence of radiation. As the facility is underground, signals from GPS satellites cannot be used. However, the tunnels are equipped with 3G/4G cellular networks. In this initial work [2], radio propagation measurements were already performed in the LHC. In this project, a radio propagation simulation will be set up using the available geometry, which will allow the 3G/4G signals in the CERNinjector complex to be used to localize individuals and propose an appropriate route to reach the selected destination.

Bibliography


Project Areas

The following are supervisors’ general areas of interest, intended to guide the formulation of bespoke project proposals. Any project proposal that emerges from these general areas will still require approval by the Board of Studies.
Area: Computer Graphics

Supervised by: Keith Bugeja, Sandro Spina

Keywords: Rendering, Realtime, Global Illumination, Geometry Modelling, Scene Acquisition, Deep Learning

Synthesis

Computer synthesised images are ubiquitously found; photorealistic rendering pervades a myriad areas and disciplines, from video games and movies to engineering, medicine and architecture. In the field of computer graphics, rendering is the process by which a virtual scene containing mathematical representations of an environment is synthesised into an image. These representations typically include geometry, surface material properties, lighting conditions and camera attributes. Physically based rendering concerns itself with the generation of images that are virtually indistinguishable from real world images. Thus, the physical behaviour of light is simulated as accurately as possible in an effort to model real world phenomena and replicate them in the final synthesised image. The essence of physically based rendering is captured by the rendering equation, an integral equation that can be used to evaluate lighting at a specific point in an environment [4]. The rendering equation cannot be solved analytically because of the complexity of the models involved. Numerical simulations based on the finite element or Monte Carlo methods are employed instead [7]. Physically based rendering is a computationally expensive process and, as is often the case with such physical simulations, parallel and distributed computing are introduced to speed up computation time or solve larger problem sets [2][3][1].
Following the real-life process by which an image sensor converts an optical image into an electronic signal, the simulation of light transport in physically based rendering calculates lighting from the point of view of a virtual camera. The interaction of light with the environment is typically classified into two types of contributions, direct and indirect. The direct contribution models light directly incident on surfaces, that is, light emitted from a source that interacts at most once with a surface before reaching the virtual camera. The indirect contribution captures light that has interacted with multiple surfaces prior to reaching the virtual camera. Lighting models that are based on the direct contribution alone are referred to as local illumination models, while models that aggregate direct and indirect lighting are called global illumination (GI) methods (see Figure 1) [5]. Local illumination models can be computed quickly and efficiently, and have been predominantly used in realtime rendering. On the other hand, GI is more complex due to the myriad of possible interactions of light between different surfaces in a scene. In physically based image synthesis, the modelling of GI is essential for physical correctness, to bolster realism. Traditionally, local illumination methods have been accelerated through the use of Graphics Processing Units (GPUs); with the advent of hardware-accelerated ray tracing (e.g. RTX-cores), new avenues have opened for point-based global illumination algorithms running at interactive frame rates.

Acquisition

The acquisition of 3D point clouds representing the surface structure of real-world scenes has become common practice in many areas including architecture, cultural heritage and urban planning. Improvements in sample acquisition rates and precision are contributing to an increase in size and quality of point cloud data (see Figure 2). The management of these large volumes of data is quickly becoming a challenge, leading to the design of algorithms intended to anal-
yse and decrease the complexity of this data. Point cloud segmentation algorithms partition point clouds for better management, and scene understanding algorithms identify the components of a scene in the presence of considerable clutter and noise. In many cases, segmentation algorithms operate within the remit of a specific context, wherein their effectiveness is measured. Similarly, scene understanding algorithms depend on specific scene properties and fail to identify objects in a number of situations. Both images (photos) and point clouds are necessary to capture different aspects of a scene. While it may be a trivial task for a person to recognise objects and structures in an image or a point cloud, this is not a straightforward task for a computer system. This is particularly challenging in scenes comprised of an unknown number of different objects, where the complexity of the identification task is augmented with the added challenge of determining object boundaries, with objects which may only be partially visible to the sensor due to inter-object occlusions.

Bibliography


Area: Runtime verification of smart contracts for blockchains and Distributed Ledger Technologies

Supervised by: Gordon Pace, Joshua Ellul

Runtime verification is a technique used to transform a specification into verification code which is injected in a system such that one obtains a version which continuously monitors for violation of the specification, hence guaranteeing that any incorrect behaviour is immediately discovered and addressed.

Smart contracts are essentially programs which executed, in a trusted manner, code affecting a network of peers. Their most widespread use is that on blockchain and distributed ledger technologies (DLTs), where such code handles and regulates the interaction between users, including transfers of digital assets such as cryptocurrency. Since such contracts may handle the equivalent on millions of euros worth of assets, their correctness is crucial, and there have been various documented cases of bugs which led to the loss of hundreds of millions.

We have recently built Contract Larva\(^1\) which handles runtime verification of smart contracts written in the programming language Solidity for Ethereum (and other DLTs) \(^1\). There are various topics of exploratory research which students may participate in at this stage, including:

- Extending Contract Larva to handle Solidity language features which are not currently covered e.g. arrays and mappings.
- Exploring alternative architectures of the tool to construct monitors in different ways.
- Applying Contract Larva to a number of real world smart contract projects.

Bibliography


\(^1\) See github.com/gordonpace/contractLarva.
Area: Static analysis of Smart Contracts using KeY

Supervised by: Gordon Pace, Joshua Ellul

The KeY system\(^2\) allows the automatic and semi-automatic verification of pre- and post-conditions of Java programs. We have recently [1] shown how it can also be used to verify Solidity smart contracts on the Ethereum blockchain platform (see the previous proposal for more information about smart contracts and blockchain).

There are various topics of exploratory research which students may participate in at this stage, including:

- Extending the translation from Solidity to Java to handle a larger subset of the Solidity language for verification.
- Applying the approach to a number of real world smart contract projects.
- Exploring concrete ways of combining runtime verification (see the previous proposal) and verification using KeY.

Bibliography


\(^2\)See www.key-project.org.
Area: The Run-time Performance of Concurrent Programs

Supervised by: Kevin Vella
Keywords: Concurrency, Systems

While the Go language’s run-time scheduler is quite fast and efficient when compared to other concurrent language run-times, the cache churn that is typically observed after the scheduler switches between thread contexts can substantially degrade application performance. Thread batching has been proposed as a way to counter this phenomenon in concurrent languages such as Manticore, occam-pi and Go itself. Moreover, the application of lock-free and wait-free techniques to streamline scheduler internals may yield additional improvements in application performance.

The Go language’s run-time system and scheduler is itself written in Go, resulting in a relatively readable code base. The aim of this family of projects is to broadly reengineer Go’s run-time system to incorporate a range of improvements, and to conduct experiments that quantify their effect on run-time performance.

Bibliography


Area: Coding Theory, Image Processing and GPU Computing

Supervised by: Johann Briffa

My general research interests involve Coding Theory (both Error Control and Data Compression), Signal / Image Processing, and High-Performance / GPU Computing.

Current topics of interest for the supervision of final year projects are listed below. While I am happy to supervise students with any background, keep in mind that suitability for a project topic depends on the student’s background (e.g. for imaging one would need background in signal or image processing). In all cases, the student needs to have a solid programming background (any language) and willingness to learn new languages and APIs as needed.

**High-Performance / GPU Computing** Generally these will involve programming NVIDIA GPUs using CUDA, for the implementation of scientific algorithms [software development, with potential for publication and further research]

**Signal / Image processing** Generally these will involve the implementation of new algorithms for light-field imaging and other next-generation applications, possibly including experimental work [some software development, research-oriented]

**Coding Theory** These are likely to involve algorithms for designing and decoding codes for insertion-deletion channels, as applicable to high density magnetic recording, DNA storage, etc. [software development, with potential for publication and further research]

**Software Development** Some specific applications for Android, Google APIs, ARM microcontrollers etc. These projects are of a software-development nature; some of these have potential for publication. These tend to be more application-oriented and proofs of concept.

Bibliography


