400-Level Project Proposals 2017

A list of possible 400-level projects is given below. You do not have to restrict yourself to this list, and can make up your own project topic. However, to do so you will have to arrange for a supervisor. You can find more information about the research interests of the staff members on the Computer Science Department Web pages.

You should email your project selections to the project co-ordinator, 400projectadmin@cs.otago.ac.nz, by **Friday 3th March**. Before making these selections you should talk to the supervisors of the projects you are interested in. In most cases the best way is to email them to make a time to do so – supervisors' email addresses are given with each project.

In order to allocate the projects as fairly as possible we ask that you give first, second, and third choice projects, and that you choose projects from three different supervisors. If you find multiple projects that look equally appealing, then you can give lists of projects as first, second, and third choices. You will still need to include at least three different supervisors, and at least two in your first two lists.

There are two different project papers - COSC480 and COSC490. The relevant Web pages (for COSC480 and COSC490) have more details, but the short version is that if you are enrolled in the Computer Science Honours degree you take COSC490, students in the first year of a Masters take either paper, and most others take COSC480. Most projects are suitable for either COSC480 or COSC490, but a few might be only suitable for COSC480, which will be indicated in the project description.

1. **Using Field Programmable Gate Arrays (FPGAs) within the teaching of computer architecture**

   David Eyers (dme@cs.otago.ac.nz)
   Andrew Trotman

   We have some FPGA boards. These devices facilitate a fairly flexible form of programmable hardware. You program a specification in a hardware description language, and the toolchain of the FPGA boards deploys it to the hardware.

   Andrew and I both have a particular fondness for the MOS 6502 CPU. We'd like you to build one to run on the FPGA boards. There are already 6502 designs on GitHub, so you would not need to start from scratch.

   Our current goal would be to make the resulting system as educationally rich as possible, so extensions to the 6502 design would probably involve additional debugging and diagnostic output. However, we would be open to other suggestions – surely there must be some sort of use for an 8-bit CPU with a 512-bit vector co-processor?

2. **Implementing an elastic CPU scavenging platform with Docker containers**

   David Eyers (dme@cs.otago.ac.nz)

   The power use of an idle computer is typically a sizeable proportion of its maximum power use. Thus consolidating as much running software as possible onto the smallest number of computers is desirable, provided that it does not cause resource contention. This has in part led to large-scale server virtualisation and thence cloud computing, as well as assisting the adoption of "CPU scavenging" schemes such as Folding@home and SETI@home. (Nonetheless, it is important not to overlook the alternative fact that the network infrastructure uses no power.)

   Apple used to provide, but has now killed off, a service called Xgrid, that facilitated CPU scavenging over a set of computers (e.g., within a teaching lab). This project aims to use Docker container technology to recreate an Xgrid-like service.

   The intention would be to build and run demonstrations that can show, for example, a web application that becomes increasingly responsive as a larger number of back-end containers spin up to "scale out" a NoSQL database that the web application is using.

3. **Web pipes**

   David Eyers (dme@cs.otago.ac.nz)

   Pipes are frequently used within shell scripting on Unix-like operating systems. They are a simple, yet highly effective inter-process communication paradigm. Shells create anonymous pipes between process invocations sequenced using the '|' character. Named pipes allow pipe stages to be connected together at a time independent from the time that they are created: the pipe endpoints sit in the filesystem. Pipes can span multiple computers, by including invocations of tools such as the Secure Shell (SSH), netcat, or wget within pipelines.

   This project aims to integrate named pipes into web browsers. This goes beyond simply using HTTP(S) for the data transport of a pipe: wget, curl and friends can already do this. The browser integration is to allow pipe creation, configuration, and local and remote pipe linking to be done with user involvement, and with the involvement of the browser's JavaScript engine.

4. **Building a scalable distributed entropy infrastructure**

   David Eyers (dme@cs.otago.ac.nz) Jim Cheetham and Paul Campbell

   Software used in computer security often requires a source of randomness, e.g., for the generation of cryptographic keys. While modern CPUs include components that sample their physical environment to derive random numbers, disclosures about the NSA and other similar organisations' interference have thrown doubt over the trustworthiness of these components.

   Jim Cheetham and Paul Campbell developed the OneRNG `http://www.onerng.info` as a USB-connected source of entropy that can be verified, both in terms of hardware and software.

   This project is to design and develop a network-based, scalable random number generator. This will be useful for cases such as cloud-hosted servers, into which you cannot plug your own hardware random number generators, but to which you are able to deliver a network stream.

   When higher rates of random data are required than can be safely provided, just plug in more OneRNGs, and the system should automatically reconfigure itself to use them. (Note the related projects Pollen and Pollenate from Ubuntu.

5. **Proving the correct behaviour for an online ballot box system**

   David Eyers (dme@cs.otago.ac.nz)

   Many years ago, I quickly hacked together some electronic ballot box software for my college student association. The Basic Online Ballot-box (BOB) has now been adopted by a number of other organisations. The ballot system is explicitly designed so that software bugs cannot cause undetectable manipulation of election results, and it has had fairly thorough code review over the years. Even so, this design and review process has not systematically demonstrated BOB's correctness.

   This project will take advantage of the small size and simple workflow of the BOB codebase to apply existing static analysis tools to prove the correctness of its implementation, or to generate provably correct subsets of its code.

6. **Reinforcement learning for scene navigation using a game engine.** Lech Szymanski (lechszym@cs.otago.ac.nz)

   Can an AI agent teach itself to navigate a 3D virtual scene using computer vision? The aim of this project is to develop and build an agent that process images using a convolutional neural network trained with a reinforcement learning algorithm. This project will involve will some game programming in Unreal game engine (C++) and machine learning using a Torch plugin (Lua).

7. **Geometric Algebra for camera position estimation from images.**

   Lech Szymanski (lechszym@cs.otago.ac.nz)
   Steven Mills

   Given two images of the scene viewed from different angles, it is possible to compute the relative position of the camera that took the photos. This problem has been solved and it involves a bit of matrix computation based on set of corresponding features identified in the two images. In this project we want to use geometric algebra, an alternate mathematical framework for computational geometry, to solve the same problem. The objective is to investigate whether the solution, stemming from different mathematical paradigm, offers any advantages/disadvantages over the established ones.

8. **Computer game to assess and train statistical competence.**

   Lech Szymanski (lechszym@cs.otago.ac.nz)
   Belinda Cridge (Pharmacology & Toxicology)

   This project aims to assess the ability of people to design studies that involve statistical analysis. The student will design and develop a computer game (most likely in Unity) that challenges researchers to solve statistical problems in a fun and engaging manner. The objective is to gather data about players competence as well as to teach and improve understanding of statistical analysis.

9. **CodeLingo for C/C++**

   Andrew Trotman (andrew@cs.otago.ac.nz)
   Jesse Meek (CodeLingo)

   CodeLingo is an AI source code reviewer that analyses your source code and identifies places where you have diverted from best practice. It is currently available for Go and PHP and extensions are underway for Python, Java, and JavaScript. In this project, you will work with this Dunedin company to extend CodeLingo to include C and C++. There are several stages to this project. First, a study of existing C/C++ code analysis tools in order to understand how these tools are currently used. Second, an analysis of the kinds of mistakes that C/C++ programmers make that CodeLingo could help with (based in the first requirement). Next, integration of an existing C/C++ parser into CodeLingo. Then, the writing of rules identified in the second stage in the CodeLingo language. Finally, running CodeLingo on a pre-existing C/C++ codebase.

10. **Deep Learning for Identifying Clouds**

    Andrew Trotman (andrew@cs.otago.ac.nz)
    David Eyers and Brendan McCane

    "Whether the weather be fine, or whether the weather be not, whether the weather be cold, or whether the weather be hot, we'll weather the weather, whatever the weather, whether we like it or not" (anonymous). One thing that will help us weather the weather is knowing what to expect. Cheap personal weather stations are readily available and can tell us the current temperature, humidity, and barometric pressure. They cannot predict the weather, and they do not tell us anything about the clouds. In this project well use a neural network (possibly TensorFlow) to learn to classify different cloud types then install a digital camera (probably a smart phone) to continuously identify whats happening in the sky. This, along with readings from the weather station will be used to predict the short-term (i.e. today's) weather.

11. **How to Stem**

    Andrew Trotman (andrew@cs.otago.ac.nz)
    Richard O'Keefe

    Stemming is the process of transforming different word variants into a common base (e.g. swimming, swam, swims → swim). Search engines use stemming to increase the quality of their results. There are many different stemming algorithms including Porter, Krovetz, Paice-Husk, and so on. In this project, we will investigate a question about stemming: How can we predict which stemming algorithm will be most effective on a document collection weve never seen before? That is, what is it about the document collection (and queries, and ranking function) that make the different stemming algorithms effective in different places?

12. **Query Performance Prediction**

    Andrew Trotman (andrew@cs.otago.ac.nz)
    Richard O'Keefe

    There are a number of ways to predict how good a search engine's results are likely to be before the search engine does the work of evaluating the query. There is some uncertainty as to whether or not these Query Performance Predictors work or not. In this project we will take a new approach to answering this question – we will use them as ranking functions. Specifically, we will use them to re-organize the top search results after the query. If QPP works then this should increase the quality of the results. If it does not then it will not.

13. **No More 10 Blue Links**

    Name (address@cs.otago.ac.nz)

    Andrew Trotman (andrew@cs.otago.ac.nz)

    Search engines famously produce a results list containing 10 results and 10 blue links to the pages. In this project, we will ask if it is possible to go beyond the 10 blue links, to a two dimensional representation of the results. Most queries contain fewer than three words, so the results could be presented as a graph with each axis representing the relevance of a document with respect to a given term. With 3-term queries, this could be done in 3D. We will take the output of a search engine and draw the results rather than listing them, and then we will test the quality of our presentation using human subjects.

14. **Approximate Nearest Neighbours**

    Brendan McCane (mccane@cs.otago.ac.nz)

    Nearest neighbours is one of the simplest classification algorithms. Simply label an new data point the same label as its nearest neighbours. It is also very effective. Unfortunately it is quite inefficient. In its simplest form the algorithm computes the distance to each of the points in the data set, and for some problems this could be in the billions or trillions. Approximate nearest neighbour algorithms attempt to maintain the advantages of nearest neighbours but do so efficiently. They trade off exact matches for approximate matches. This project will look into various approximate nearest neighbour algorithms for a diverse set of problems.

15. **Data mining first year programs**

    Brendan McCane (mccane@cs.otago.ac.nz)

    We have a database of COMP150 submissions for mastery tests that span 3 years now (since these tests were introduced). It would be very interesting to analyse these submissions to see how we can better teach the students who are struggling. This project will involve developing methods to automatically analyse and cluster the submissions with the aim of trying to figure out what it is that students struggle with most.

16. **Distinctive Features**

    Brendan McCane (mccane@cs.otago.ac.nz)

    Object recognition often involves identifying objects based on their overall characteristics (e.g. the distinctive shape of a banana). Humans are able to not only recognise based on overall characteristics, but also to describe unusual characteristics in an object (e.g. that banana is pink). This project will look at ways for an object recognition system to focus on these unusual characteristics.

17. **Robot Learning**

    Brendan McCane (mccane@cs.otago.ac.nz)

    We have a robot arm and can attach cameras to it so it can see. In this project you will implement systems to allow the robot to learn how to control itself and focus on interesting and new objects in its field of view

18. **Computing with light**

    Brendan McCane (mccane@cs.otago.ac.nz)

    LightOn (`http://www.lighton.io/`) is a company that uses light to perform specific computations of interest to machine learning. Either through simulation, or the help of someone in physics, this project will investigate this technology and see what else could be done. A background in physics and/or maths would probably be helpful.

19. **Modelling Swimming Motions**

    Brendan McCane (mccane@cs.otago.ac.nz)
    Chris Button and Nigel Barrett (Physical Education)

    Researchers in Physical Education are making miniature waterproof motion sensors to attach to the wrists, ankles, and torso of swimmers. The aim of this project is to analyse the output of these sensors to identify the relative phase (co-ordination) of people's limbs when treading water or swimming in the flume (a circulating water channel).

20. **Computational videography**

    Stefanie Zollmann (stefanie@cs.otago.ac.nz)
    Steven Mills and Tobias Langlotz (Information Science)

    Computational videography is the extension of traditional videography using computing technology, such as image processing. For the related field of computational photography, a lot of different aspects have been already explored such as using segmentation and seam carving for content-aware image resizing or using pose estimation from old and new photographs for computational rephotography. For computational videography, this is getting more challenging due to the increased amount of image data, requirements of temporal coherence and dynamics in the image data. In this project, we will investigate the options that arise from using real-time computer vision and computer graphics for computational videography.

21. **The Augmented Reality Tutor**

    Stefanie Zollmann (stefanie@cs.otago.ac.nz)
    Steven Mills, Tobias Langlotz (Information Science)

    Augmented Reality (AR) allows displaying digital content, such as annotations, directly attached to real-world objects. This way of presenting information reduces the mental effort of mapping information to an object of interest. This makes AR a powerful tool for learning environments where often information from a book or computer has to be mapped to real objects (e.g. specimens in Anatomy). In this project, we will investigate different ways of information presentation and information linking for learning environments. In addition, we want to address the challenges that arise from this way of information presentation, such as annotation presentation for X-Ray visualisation.

This project also involves collaboration with Denis Kalkofen (Graz University of Technology) and Markus Tatzgern (Salzburg University of Applied Sciences).

22. **Indirect Augmented Reality Browser**

    Stefanie Zollmann (stefanie@cs.otago.ac.nz)

    Low-cost built-in sensors (e.g. GPS, compass and gyroscopes) such as available in mobile phones usually come with large positioning and orientation errors. This is a general problem for creating high quality Augmented Reality (AR) experiences where accurate alignment is required. Indirect Augmented Reality uses preregistered image data (e.g. panoramic images) to create AR overlays. This is an option for addressing the registration problem for AR browsers and for achieving stable and sufficiently accurate AR visualisations on mobile phones. In this project, we will integrate pre-registered image data from publicly available databases and combine this with GIS data to create an Indirect AR browser for pedestrian navigation and sightseeing and investigate the feasibility of such Indirect AR browsers.

23. **Visual Computing for Sports**

    Stefanie Zollmann (stefanie@cs.otago.ac.nz)

    Professional sports training, sports performance coaching, as well as sports entertainment often require a detailed analysis of the events happening on the sports field. Computer vision and computer graphics have the ability to contribute to a more automated analysis in various ways. In this project, we will research the possibilities that arise from using Visual Computing, such as computer vision tracking within sports environments. We will also investigate the processing of such information for an adequate presentation. This project will suit students that have an interest in computer vision and computer graphics.

24. **Language Identification**

    Richard O'Keefe (ok@cs.otago.ac.nz)
    Andrew Trotman

    Language identification means taking a document, presumed to come from a known set of languages, and deciding which language it is from. This idea can be applied to both human languages like Spanish and Māori and to programming languages like Java and Python.

    We had a 4th-year project on this a few years ago and got some good results. RO'K has some ideas on how to do better, and would like to test them. We should also explore different document sizes: it's obviously harder to classify a single word than a whole book.

    We have two multi-lingual document collections and can get another. We also have a collection of hundreds of programs written in dozens of programming languages.

    This involves straightforward programming with text, except that you have to handle a lot of it.

25. **Sensor compression**

    Richard O'Keefe (ok@cs.otago.ac.nz)
    David Eyers

    The department has done a lot of work with tiny computers with several sensors communicating by radio. Sadly, radio communication costs a lot more energy than computing, so we want to reduce the amount of transmission as much as we can. This is important in the growing field of Precision Agriculture, for example.

    Much of the work on sensor networks is application-independent. But weather sensors may be fairly coarse, weather doesn't change fast all the time, and nearby stations experience similar weather. Also, if an application does not data every second, or does not need great precision, we might be able to get by with less data less often.

    This project will explore an application of your choice (evaporation rate prediction is suggested) that uses weather station data (such as wind speed, temperature, pressure, humidity, and others) from several stations. The aim is to see how low we can get the transmission rate while maintaining acceptable quality.

    This project will be based on simulation, not actual devices.

26. **Micropatterns**

    Richard O'Keefe (ok@cs.otago.ac.nz)
    David Eyers

    You have probably heard of Design Patterns, like Observer and Composite. There is published work on Micro-patterns, focusing on Java. How much of that can be applied to other languages, and if it's applied to other languges, is it useful?

    You could for example write or adapt a parser for another OO language to detect micro-patterns, and try to extend the catalogue with patterns for your chosen language.

    Or you might mine revision control logs for selected open source projects to see which, if any, micro-patterns are associated with need for revision.

    This project is all about processing program source code as data.

27. **Latent Semantic Mapping**

    Richard O'Keefe (ok@cs.otago.ac.nz)
    Andrew Trotman

    "Latent semantic mapping is a data-driven framework to model globally meaningful relationships implicit in large volumes of (often textual) data. It is a generalization of latent semantic analysis. In information retrieval, latent semantic analysis enables retrieval on the basis of conceptual content, instead of merely matching words between queries and documents." – Wikipedia.

    Mac OS X has had library support for latent semantic mapping for some time, so the amount of programming required to use this technique is relatively small.

    The goal of this project is "do something interesting with Latent Semantic Mapping". For example, given a collection of hundreds of programs in hundreds of languages, what might we find? What might we find in the (freely downloadable) JavaDoc for Java? Can we use this to make a tool that says "looking at the class you just wrote, here are some existing related classes you might be interested in"?

28. **Having fun with programmable hardware.**

Richard O'Keefe (ok@cs.otago.ac.nz)
David Eyers

A Field Programmable Gate Array is reprogrammable hardware. You can write a hardware design in a high level language (the two main ones are VHDL and Verilog) and compile it, down load it into an FPGA, and run it. VHDL and Verilog can also be compiled to silicon, to make application specific integrated circuits, so FPGAs are often used to prototype things. One can design and implement ones own CPU, but it is more useful to implement an algorithm directly without any instruction interpretation overhead.

We have a couple of FPGA boards and some textbooks and free software.

It would be particularly interesting to see if these things can be used in information retrieval. For example, the inventor of the widely used Porter stemming algorithm has devised a special purpose programming language for writing stemmers; is it possible to translate stemmers into VHDL so they can be compiled into hardware? Even more basically, Unicode is enormously complicated, even a simple thing like asking "are these two strings equal" is hard. Can we implement that efficiently on an FPGA?

If you have something of your own you'd like to try, that would be welcome too.

29. **Latent Methods**

Richard O'Keefe (ok@cs.otago.ac.nz)

I've recently come up with an idea called "latent methods". These are methods that you define in some class, but they do not become available until a subclass implements the interface(s) they are declared to need. This project would involve

(a) critiquing the idea and comparing it with other approaches like multiple inheritance,

(b) implementing latent methods in some OO programming language, possibly by modifying a compiler or writing a preprocessor, and

(c) writing test cases for your implementation that can be used as examples to show the usefulness (or otherwise!) of the idea.

30. **Semantic Relationships in Source Code**

    Richard O'Keefe (ok@cs.otago.ac.nz)

    Some programming languages, notably C#, Java, and Smalltalk, have a standard way to attach "annotations" to code. This is usually used to provide the compiler and run-time system with more information like "this reference should not be null", "this is a test case", or "this is part of an Object-Relational Mapping". The nearest we tend to get to something for the programmer's edification is "don't use this any more".

    But chunks of source code are related to each other and to external documents in ways that the compiler and run time system do not need to know about, while maintenance programmers would love to know. The idea is to take an existing body of code that you are not familiar with, read it, and note what kinds of links are either informally specified or entirely missing but important to you, so that we can build up this catalogue. You should also review what has been done in Java and C# and Smalltalk. For more work, you might construct an actual set of annotation definitions that Java (or C# or Smalltalk) programmers might use and provide some tools (perhaps by extending an existing IDE) to exploit them.

31. **Explorations in Rosetta Code**

    Richard O'Keefe (ok@cs.otago.ac.nz)

    www.rosettacode.org is a growing collection of problems (currently 780) and solutions to them in hundreds of programming languages. It has been suggested that this is a useful resource for Software Engineering analyses. Since the site was not developed for that purpose, it is not surprising that there are some questions. For example, the size metric used to compare solutions is very weak; for C it can vary by 28% depending on the options you give to a code reformatter. Is it similarly weak for other languages? It is known that some problems are interpreted more than one way, making solutions incomparable; how common an issue is this? The 8 languages compared do not have solutions to exactly the same problems; are the conclusions of previous work still sound if we add more languages or more solutions?

    This is a fairly open-ended introduction to software engineering research and code mining that should lead to a publication.

32. **Fun with Parallela**

Richard O'Keefe (ok@cs.otago.ac.nz)

I have a Parallela board. This is a pocket sized board with 18 cores: 2 ARM cores running Linux and 16 RISC cores, using less than 5 Watts and costing about $150. Programming this beast isn't like programming a conventional multicore CPU (although you can use OpenMP) or a GPU (although you can use OpenCL) because the 16 RISC cores have small memories with fast interconnects. The point of the project is to develop working software that does something interesting in parallel and to measure performance relative to a multicore CPU or a GPU.

"Something interesting" might be some sort of combinatorial optimisation algorithm (other than travelling salesman), a genetic programming package, or a data mining task. If you have another idea that would be wonderful.

If your code is clear and well described, the company that make the board might be happy to display it on their site.