COMPSCI 715
Advanced Computer Graphics

Using Related Work

Today's Mission

1. How do you find good related work for your project?
2. How does a good related work section look like?
3. What have related works contributed to your research problem?

Typical Research Paper Structure

1. **Introduction**: What is the research problem? Introduce and motivate it. Summarize your contributions.
2. **Related Work:** What have others done? How is it different? Cite, summarize other solutions & compare it with your own.
3. **Design:** Your solution. Describe it in enough detail so others can implement / replicate it. Software architecture (e.g. class diagram)? User interface (e.g. screen diagram)? Algorithms?
4. **Implementation:** How have you implemented your solution? Tools and technologies used? Implementation challenges?
5. **Evaluation:** Explain the methodology you used for evaluation. Present the results. Discuss them.
6. **Conclusion**: Summarize contributions. Point out future work.

Some General Writing Tips

- Use direct and simple language
  - Focused: If it is unnecessary, take it out, e.g.
    - *Bad: "This paper tries to build a system that could do X"*
    - *Good: "We do X"*
  - Active voice better than passive voice, e.g.
    - *Bad: "The system was build by us by extending X"*
    - *Good: "We built the system by extending X"*
  - Compact: Break long sentences into shorter ones, e.g.
    - *Bad: "A improves B, but while A needs C, D doesn't, so D is a good choice."*
    - *Good: "A improves B, but needs C. D doesn't need C, so it is a good choice."*
- Form logical units with a logical flow
  - Each section has a clear purpose - don't dilute it
  - Each paragraph describes one idea / argument - don't mix
  - If you need A to understand B, then A comes first

What are Contributions?

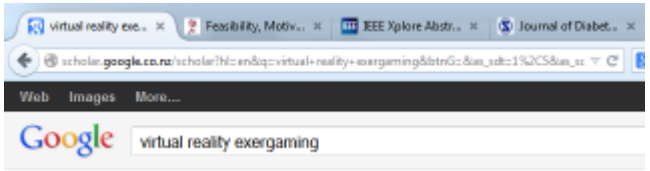
CC Rob Lavinsky - iRocks.com

The “gold nuggets” of new knowledge in a publication

- Novel: nobody has done it before, reasonably non-obvious
  - Argue with related work: explain what others have done (their contributions) and what they haven’t
- Useful in some sense - applicable to other projects / problems
  - See motivation, e.g. in the introduction
- Scientific:
  - Clear: understandable, reproducible
  - Right level of abstraction: provide the interesting and not the boring details
  - Right terminology: write like other researchers
  - Evidence-based: convince the readers with facts

Finding Related Work

1. Gather papers that might be relevant
   - Keyword search with different words (e.g. Google Scholar, ACM, IEEE)
   - Snowball search: follow up the references (cited and citing papers)



Astrojumper: Motivating exercise with an immersive **virtual reality exergame**
S Finkelstein, E Suma - Presence, 2011 - ieeexplore.ieee.org
Abstract We present the design and evaluation of Astrojumper, an immersive **virtual reality exergame** developed to motivate players to engage in rigorous, full-body exercise. We performed a user study with 30 people between the ages of 6 and 50 who played the ...
Cited by 16 Related articles All 6 versions Cite Save More

2. Filter papers that are actually relevant by reading the abstract
3. Read the papers
   - First just scan over it (figures? sections? methodology?)
   - What is the research problem? What are the contributions?
   - Do they cite related work that is useful for you?
   - See http://blizzard.cs.uwaterloo.ca/keshav/home/Papers/data/07/paper-reading.pdf

Peer-reviewed Publications

- Scientists submit papers to journals or conferences that are peer-reviewed
  - Short paper ~5 pages, long paper ~10 pages, journal article ~20 pages
- Other scientists, established experts in the field, review the papers, free of charge
  - Individual reviewers remain anonymous, but the board of possible reviewers is known
  - They check for: obvious inconsistencies, dubious statements, good standards of presentation, sufficient degree of completeness, i.e. disclosure of details, and novelty of the results
  - They usually will request changes
- Many papers get rejected (usually 50% or more)

Citing Publications

- Cite quality assured publications using proper citation style
  - Peer-reviewed (not just pro forma, e.g. no fake conferences)
  - Cited by others in the field (means you should know about it if you are working seriously)
  - Well-known conferences & journals (e.g. see rankings)

- If you need to refer to a webpage, use footnote with URL instead of citation
- Industry white papers and reports can be cited, but are not quality assured, so no strong backup for you (“grey literature”)
- Textbooks only for very specific things (refer to them with page or chapter number)
- Not Wikipedia (good for your own overview, but unreliable)

Writing about Related Work (~1 page 2-column)

It is all about the contributions and showing how yours are novel:

1. Summarize the contributions of the good related works in one paragraph per publication
2. Organize the paragraphs by grouping them in subsections
3. Compare the related works with your contributions to show how your contributions are novel
    - What are the similarities to your work? Give credit where it is due.
    - What are the differences?
        i. Is yours new knowledge?
        ii. Is yours better in some way? E.g. more powerful, simpler, less limited

Exercise

Learn from good and bad example introductions

1. Read the related work section (note: examples are shortened)
2. Identify the following parts:
   a. Sources for the related works
   b. Contributions of the related works
   c. Differences to own contributions of the project
3. Discuss:
   What are the good and bad points?
   How could it be improved?

Good Related Work Example 1

https://www.cs.auckland.ac.nz/~lutteroth/publications/VanDykEtAl2012-GLDebug.pdf

gDEBugger[4] was one of the first commercial graphics debuggers to become widely available in 2004. It demonstrated many of the features seen in modern graphics debuggers, such as [...] The contribution of gDEBugger is in its pioneering of graphics debuggers in the commercial space, as well as [...]

GLSLDevil[9] [14] is a tool specifically aimed at debugging the shader pipeline of OpenGL applications. GLSLDevil provides novel features in that it automatically instruments OpenGL shader code. The instrumented code then outputs extra information that can be used for debugging.

GQL (graphics query language) was created along with a debugging system by Duca et al. [5]. Similar to GLDebug, it enables tracking and logging the state and calls made by an OpenGL program over the course of execution. However, the historical information is only made available through an SQL-like language (GQL) that users have to learn, and there is no direct support for comparing states and highlighting of state differences.

Good Related Work Example 2

https://www.cs.auckland.ac.nz/~lutteroth/publications/PenkarLutterothWeber2013-NavigatingHypertextWithGaze.pdf

Considerable effort has been invested in finding efficient click alternatives. The most straightforward one is considered to be dwelling or fixating on a clickable area. [...] This click alternative primarily suffers from inadvertent clicking, a problem that can be mitigated if it is possible to place the content or labels outside the clickable areas [9].

Research has also focused on compensating for the inaccuracy and imprecision of gaze tracking. MAGIC is one such example in which the user relies on explicit commands using other input devices, e.g. the mouse, while also benefiting from the speed of pointing with gaze [15]. This approach moves the mouse pointer quickly to the area being gazed at, but relies on the mouse for finer adjustments and clicking.

The EyePoint [19] is another possible solution to cater for lack of accuracy and precision in gaze tracking. [It] involves magnification (of the area being gazed at) on the press of a keyboard button. In the magnified view, the key can be released while dwelling at the object of interest, resulting in an action on that object [...] However, these techniques either block or distort the screen content, and loss of contextual information can be inconvenient and problematic, especially for tasks involving visual search [20].

Good Related Work Example 3

https://www.cs.auckland.ac.nz/~lutteroth/publications/ZeidlerEtAl2013-AucklandLayoutEditor.pdf

Rockit [16] automatically proposes constraints based on the “gravity field” of other objects. The user can select the desired constraints from a set. This is similar to previous work where constraints are inferred by snapping graphical objects relative to other objects [11]. In ALE widgets can be snapped to others in order to set up the corresponding constraints. However, ALE adds more powerful edit operations, such as as inserting a widget between others [...]

Bramble [7] connects objects using a set of interactors, which establish non-linear constraints. While this can be used to prevent overlap, the user has to add interactors manually. ALE adds such constraints automatically. [...]

Adaptive document layout [12] in the print and web domains is somewhat similar to GUI layout. However, the flow of a document constrains placement differently than a GUI. Documents typically arrange text and images in a sequential manner, and support more flexibility in the layout using algorithms for figure placement, line-break, and pagination. Grid-based [15] as well as constraint-based [3,13] methods have been used for document layout.

Bad Related Work Example 1

The various documents I read helped me to get a clear picture of what I need to achieve in the time period of completion of the project. Various links on the Wikipedia website regarding Sculpting, Subdivison surfaces and Collaboration helped me understanding the basics for starting the project [...]

The available sculpting tools already have become popular because of their completeness for the features provided which are easy for a user to understand and implement. The tool which we/I took as reference is Sculptris. A better definition than just telling it as a sculpting tool is “Sculptris is an elegant, powerful and yet easy to use 3D sculpting software, allowing the artist in you to simply focus on creating amazing 3D artwork.

3D Collaborative Sculptor aims to implement all the features similar to the features available in Sculptris. Well, might be because of time constraints of the project, it will implement less features compared to Sculptris. Our main aim is to implement the Collaborative feature in the 3D Collaborative Sculptor extending a better solution for the 3D and Graphics related companies as collaboration speed up the process of development with accuracy.

Bad Related Work Example 2

An abstract syntax tree (AST) is a tree representation of the abstract syntactic structure of source code [3]. Each node of the tree represents a construct that occurred in the source code. Abstract syntax trees are widely used in compilers, due to their ability to represent the structure of the program's source code. AST is usually generated after the syntax analysis phase of a compiler.

Design Patterns are known solutions to common design problems in software engineering [13]. They are well known solutions to general design problems. Design patterns are usually defined as a relation between the interacting objects of a software system or the way classes are structured in the source code. Anti-Patterns are bad coding practices in oppose to Design Patterns.

Code Smells are also bad coding practices but they are different to Anti-Patterns, they are usually code taints such as long methods, code duplication and data classes. Code Smells are tend to be local code taints within methods or classes whereas Anti-Patterns are usually structural problems.

Bad Related Work Example 3

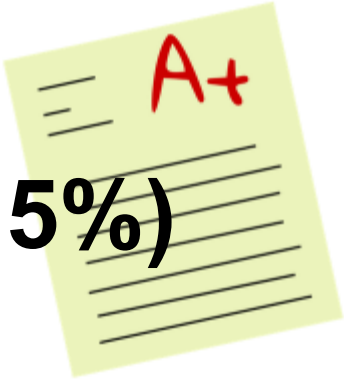
Murphy-Hill et. al. [7] worked on developing a code smell detector called Stench Blossom which offers an interactive visualization tool to give programmers a conceptual overview of the smells in source code and helps them in comprehending the sources of the code smells.

Neukirchen et. al. [8] worked on developing an anti-pattern detecting system which nds anti-patterns in test suites. Their open-source refactoring and metrics tool named TRex was capable of automatically detecting code smells.

Stoianov et. al. [10] used a logic-based approach to detect anti-patterns in source code. Their solution uses an infrastructure named Jtransformer that is an Eclipse plugin which generates logic facts that is a representation of java source code.

Emden et. al. [11] have worked on implementing a system that visualizes code smells. They implemented a prototype code smell browser called jCOSMO for the detection and visualization of code smells in java source code.

This Week's Assignment: Write Related Work Section (2.5%)



Write a related work section for your project
(~1 page double-column)

- Individual submission, no group work
- Worth 2.5% of your final mark
- Again, methodology and contributions are hypothetical: imagine your project is over and was successful
- Be professional: try to imitate well-written introductions
- Use LaTeX, e.g. https://www.writelatex.com/

Submit PDF by Sunday 17/8 7pm to assignment dropbox:
https://adb.auckland.ac.nz

All the best :-)