NewsLetter

ISPRS Student Consortium

Issue No.1, Vol. 10 - June 2016

SPECIAL ISSUE

Open Source Software Part-2

Interview with Mr. Tomoji Takasu, developer of RTKLIB

Python for Scientific Computing

Reference management with Zotero

isprs

ISPRS SC NewsLetter

SC Newsletter (ISSN Y506-5879) is published
by ISPRS Student Consortium.

To join our members area visit
www.isprs-sc.org

Editor-in-Chief:
Ivan DETCHEV

Technical Editor:
Ayda AKTAŞ

Proof-Reading:
Sylvie BROWNE

Contributors:
Ivan DETCHEV
Jeremy STEWARD
Hiroyuki MIYAZAKI
Urša KANJIR
Adam CLARE
Ayda AKTAŞ

Frontpage designed by Ayda Aktaş

Would you like to join SC Newsletter team? Do you want to make a difference? Want to learn new skills?

SC Newsletter is at a stage where getting broader and better demands more people to be involved in the process of it's formation. That's why SC Newsletter team is looking for the following volunteers:

- More **people who would be willing to prepare articles** for existing or new rubrics,
- Designers of Newsletter

If you can help us with any of the above, please let us know!

info@isprs-studentconsortium.org

And also...
If you **would like to publish your research work** in the SC Newsletter send us your abstract on email written above. We will soon contact you for further information.

Dear ISPRS SC Newsletter readers,

There is a rapid growth in the popularity of geoinformatics in conjunction with advances in computing technology, hardware and software. As the amount of geographical data available from different sources (remote sensing platforms, surveying methods etc.) gets bigger, the need for low-cost software to describe, analyze and diagnose this data is becoming more important, especially for educational research and decision making in geoinformatics.
Nearly every student or young scientist comes to a point where open source software is a way to solve this problem. Our special issue, dedicated to open source software in geoinformatics, will give you a glimpse of this world, with different point of views from users to developers within various projects. As the newsletter team, we believe that the articles you'll read will give you many insights and maybe help you with your research.
The XXIII ISPRS Congress, when the new board members will be elected, is getting closer. We are encouraging our readers to send in their application forms if they want to have a front-row seat at ISPRS-SC.

Ayda Aktaş
ISPRS SC Publication Responsible

Table of Contents

Interview
by Hiroyuki Miyazaki

Interview with Mr. Tomoji Takasu, the developer of RTKLIB

In this article, I would like to introduce Mr. Tomoji Takasu, a researcher on GNSS technologies at Tokyo University of Marine Science and Technology (TUMST). He is the author of RTKLIB (http://rtklib.com/), the most recognized open source program package for standard and precise positioning with GNSS (global navigation satellite system). RTKLIB supports various positioning modes for both real-time and post-processing. RTKLIB also supports many GNSS and GNSS receivers' proprietary formats and protocols. In addition, it supports various types of external communication. While RTKLIB is equipped with APIs for application development, it also has several utilities to handle GNSS data processing so that non-developer users can use this program.
Through this interview with Mr. Takasu, I would like to deliver important viewpoints and offer students ideas about career development through free and open source software.

Could you give us a brief history of the development of RTKLIB?
Before the job at TUMST, I had been working as a software programmer in a firm and also as an individual software engineer for 15 years. About 7-8 years ago, Prof. Yasuda, who is a leader in GNSS science and technology in Japan, invited me to research and teach at TUMST. I joined GNSS research groups at TUMST and was given opportunities to lecture and work on leading research topics in GNSS at the graduate school level. With my experience as a software programmer, I developed some small sample programs as practical exercises for students. This was the very first version of RTKLIB.
I wanted to promote greater use of GNSS technologies and applications, so after adding several general use functions to this program, I decided to publish RTKLIB as free and open source software on my website. This was two and a half years after the first version. I was concerned that users would send me a lot of claims and complaints, but instead I found that users sent feedback which contributed to improvements in the software. The number of downloads is 10,000 for version 2.4.0 and 80,000 or more for version 2.4.2. Currently, the latest version is available at GitHub (https://github.com/tomojitakasu/RTKLIB/). Since version 2.4.0 covers almost all of the functions required for GNSS processing, I have been less active in the development of newer versions of RTKLIB.

Since RTKLIB is so highly functional, it could have great potential as a commercial product. Why have you not attempted commercialization of the software?
When I have given presentations on RTKLIB, some business persons have offered opportunities to commercialize RTKLIB. But I reject any commercialization because commercialization would require customer support operations which would interrupt my R&D of GNSS technologies. I really do like R&D on GNSS.
R&D of free and open source software is mostly motivated by increase in the number of users. However, such increase is not necessary to me. My interest has been in lowering the cost of using RTK and in broadening its usefulness. As a result, RTK is getting a solid reputation in several industries, such as precision agriculture.

What kind of users have you targeted in R&D of RTKLIB?
The original purpose of RTKLIB was for education. However, as RTKLIB has gotten popular in the GNSS community and in industry, I have developed the software to be useful for all kinds of purposes. At least, I can say that I don't target users who need detailed support in their use of the software.

How and by whom is RTKLIB used?
In Japan, there are practical applications of RTKLIB in precision agriculture, meteorology (GPS meteorology), and construction. RTKLIB is used internationally by geospatial companies developing firmware for RTK and UAV flight controllers. U-blox, a popular multi-GNSS device, also uses RTKLIB as firmware.
Although RTKLIB had been under GPL, the license was changed to BSD with version 2.4.2 for greater flexibility in business and industrial use.

Please tell us about representative projects using RTKLIB.
The most representative is the MADOCA project[1] , which has been conducted by the Japan Aerospace Exploration Agency (JAXA) for five years. RTKLIB is used in MADOCA for processing GNSS data, because RTKLIB recognizes almost all GNSS data formats especially RINEX, and for data input and output.
A second example is a project by the Geological Survey Institute of Japan for multi-GNSS application in surveying jobs. RTKLIB was used for antenna calibrations. The project released a software package, GSILIB[2] , for users' evaluation of multi-GNSS applications.
A third is an application for development of an engine to analyze and assess survey results. In this project, RTKLIB was used for research on algorithms.

Fourth is educational programs. My colleagues and I have conducted a series of international educational programs with the Institute of Positioning, Navigation and Timing of Japan (IPNTJ). We invited foreign students to complete practical exercises using RTKLIB to learn the basics of GNSS. Exercises include drawing graffiti by GNSS trajectory and maritime cruising with real-time GNSS. The programs were then provided to Japanese colleagues of the IPNTJ.

How has the community of RTKLIB grown?
GitHub is functioning well for developing the community. For example, the "Issue" function of GitHub is working as a bulletin board where users and developers can communicate and exchange ideas.
There are several local communities of RTKLIB in the world. For example, I found a German RTKLIB community, in which people have conversations in their local language.
FOSS-GPS[3] also has drawn attention to RTKLIB as an outstanding software for GNSS processing.

I think you have developed new networks and communities through RTKLIB. Please give us your thoughts in detail about it, especially regarding development of your career as a researcher and/or a specialist.
I have been offered a lot of jobs because of RTKLIB. This would not have occurred if I had not released RTKLIB to the public as free and open software. Actually, one of my current jobs is with a private company which connected with me because of RTKLIB. Academic meetings and conferences are also important opportunities to develop my career and networks.

Please give your advice to students who are willing to work on software development, especially those relevant to GNSS.
You should know the difference in the requirements of software for enterprise products versus academic research. Enterprise software requires quality control in fixing bugs, maintenance, and documentation because businesses aim to develop larger markets for their products. On the other hand, academic research requires originality and novelty through rapid prototyping more than quality control. Actually, software for academic research is not usually applicable to practical problems, so such software cannot have the sustained popularity of RTKLIB. I would like students to have intensive experience of both the very basics in academia and on-the-job training in business fields.

End...

Interview
by Urša Kanjir

Interview with Fabian Fassnacht

Can you introduce yourself?
Hi, I am Fabian Fassnacht. I studied forestry sciences in Freiburg and I am currently with the KIT in Karlsruhe. I specialized in remote sensing during my PhD. My favorite color is green.

Which software do you use or teach to your students?
I mainly use R and QGIS in my day to day scientific life. On occasion I will add a bit of Orfeo Toolbox and specialized LiDAR software. When Sentinel data became available, I also started working with the Sentinel Toolboxes (used to be PolSARpro / BEAM box).
For teaching I basically use exactly the same software as for research. I consider this a win-win situation: For me it is more economical as I don't have to deal with software that I typically do not use and the students do not learn things that will not be of great use for them when they want to write a PhD later.

How do you select which free software is going to be used for the lectures?
Ideally the software is useful for many things and not too focused on the special topic I am teaching personally. We have had good experiences at our institute using R in many lectures for one of our BSc degrees. By being confronted with the software in several courses, students - at some point - typically overcome their fear of "programming/scripting" and start enjoying it.

Why you think using free software is important, especially for academia?
Free and especially open-source software is extremely important as it guarantees independence from tight budgets – which is also very relevant for countries doing less well financially - and it supports creativity as each user can contribute something to the community. Furthermore, students are more independent and have the chance to learn without being physically bound to some computer with a certain software license that is situated at the campus.

Do you see any disadvantages in using free software?
I don't see any real disadvantages. Of course there are sometimes troubles with version updates that suddenly confront you with scripts that used to run well but decide to stop running the moment you are standing in front of 25 students and just after you had explained to them all the advantages of using open-source software. But with a bit of experience, these caveats are known and can be handled quite easily.

Python for Scientific Computing

By Jeremy Steward, University of Calgary

Much of the work we do in Geomatics today involves processing and sorting through heaps upon heaps of data. Often, the quality of the tools we use can vastly influence the ease with which we solve problems; beyond that, it can even contribute to how we frame our problems, or guide the decision process when we are unsure of what problems we want to solve. In short, and this should sound obvious, better tools can help guide us and enhance our problem solving capabilities. It goes without saying then, that we want to avoid bad tools when possible; however, it can be difficult to determine what constitutes a “bad” tool.

The common software-based tools we use to solve problems in our department are typically and traditionally C++ and MATLAB. C++ has its roots in the C programming language, and as such is known for its speed, but also for its verbosity, and well, its quirks. MATLAB, on the other hand, is a programming language from the folks at MathWorks, with its own history and its own set of baggage. MATLAB is commonly hailed for the comprehensive set of toolboxes that come with most licenses. However, that’s really the big issue: MATLAB’s licensing is proprietary, and this can cause a lot of headaches both for those with and without access to the language.

Instead, I’d like to argue why the Python programming language is a better fit for scientific computing. It balances the strengths and weaknesses of a lot of other languages, and carries a significant benefit for those who take the time to learn and eventually master its basics. Most importantly, Python code is written for the reader. Python allows you to not only quickly express your ideas in code, but remains highly readable and reusable.

Why not C++?

Certainly an established language in its own right, C++ is often a first choice amongst students and researchers when dealing with large or complex datasets that require heavy processing. While the raw speed of the language, or the availability of Free and Open compilers is hard to refute, in many ways C++ hampers our ability to develop applications or processes interactively and with minimal expressive overhead. Needless to say, the language can get very verbose, and can often require you to spend nontrivial amounts of time or effort to do what are basic tasks in higher level languages.

There’s no doubt that the C++11 and C++14 standards have improved the situation; however there’s still something lacking. Some shortcomings could be listed as follows:

- Long compile or linking times hamper interactivity.
- Third-party libraries are hard to find, and can prove difficult to integrate.
- Visualization requires specialized libraries that must compile on your architecture.
- Data exploration is hampered by the write -> compile -> run feedback cycle.

Beyond these reasons, generic programming requires heavy use of template metaprogramming, pre-processor macros, or complex class hierarchies. Compared to languages like MATLAB or Python, it’s difficult to just load your data into the interpreter and get to the essence of your problem without worrying about the concrete details of how you implement the scaffolding of your desired result.

Why not MATLAB

In contrast, many scientists and researchers flock to MATLAB because it solves some of the problems outlined above. MATLAB does have some great things going for it, namely that it is easy to load and interact with data, visualization is a breeze, and the toolboxes that come with the language can be quite comprehensive at times.

Yet, as I mentioned earlier, MATLAB contains some surprises of its own. The language suffers in part due to the licensing structure that MathWorks provides. Unlike many other mainstream languages today, MATLAB is not Free Software / Open Source, nor is it permissively licensed. Certainly, well-funded universities can pony up to purchase copies of MATLAB; however, not all researchers are well-funded, or will always have such licenses available. With MATLAB, you can’t be sure you’ll even be able to run code tomorrow that you wrote today. In the short-term, this may not have great effect, but over time this can compound significantly.

Even without the licensing issues, MATLAB has a series of issues that limit the expressivity of the language, and limit your ability to work with the language to produce results. In short:

- Third-party libraries hardly exist (some are available via MathWorks, but few are well-tested or widely-used).
- The type-system is an uphill battle if you aren’t using array or matrix types.

Hash-tables (dictionaries), structs, trees, etc. are all unwieldy or don't exist in the language as is.

- Object-Oriented Programming is cumbersome and exhibits many surprising gotchas.
- Code is difficult to organize, as there is no real "module" system.

Python's main strengths

The first thing many notice about Python, if nothing else, is the expressivity of the language. What I mean, is that with Python, you can often write and read your programs just as you would in English. More generally, expressivity can be summarized through the following quote:

Intuitively, if every program that can be written in language A can also be written in language B with only local transformations, but there are some programs written in language B which cannot be written in language A without changing their global structure (i.e. not with just purely local transformations), then language B is more expressive than language A. -- Matthias Felleisen (1991)

And in many cases, programs in Python cannot easily be rewritten in other languages, at least not without writing vast amounts of extra code. Python gets out of your way when you need it to; lends itself to learning and collaboration; and enables fast, iterative development. In research, this is the key element. Being able to quickly translate your thoughts into code becomes a unique advantage, something to drive your research and hypotheses into a tighter feedback loop.

How does Python do this? I think it comes down to a few strengths of the language and its community that outperform other, more traditional languages like C++ or MATLAB.

Dynamic, yet strong

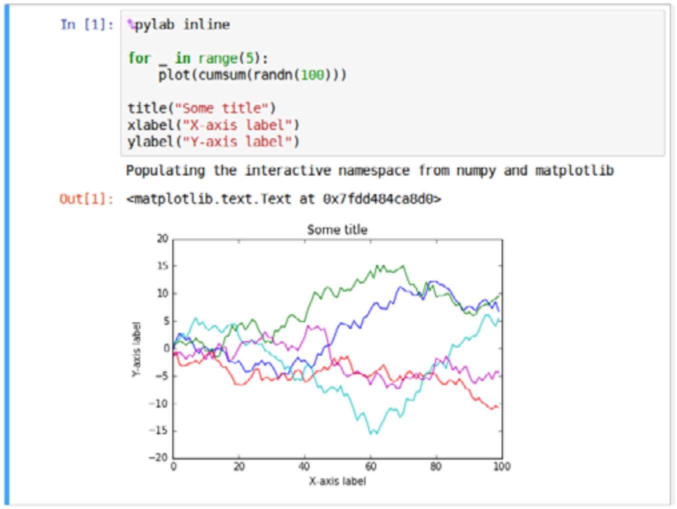
One of the main gripes many users have with C++ is the complex type system. Especially in cases where some generic types are necessary, templates can confuse and obfuscate the underlying structures or goals of the program. Perhaps the largest complaint I would raise is that often in research we don't begin programming with a complete understanding of our problem or the solution. This makes dynamic typing in a language like Python a boon, because we don't have to worry so much about the structure of each individual type as much as we focus on getting the process correct. The flexibility afforded by a dynamic type system is in many ways invaluable if you are implementing new research, as many things may change quickly as you explore and grasp a better understanding of the problem domain.

In contrast, we find that not all dynamic typing is equal. To exemplify this, we can examine MATLAB's type system. MATLAB attempts to cram everything into the idea of "arrays." Structs are considered scalar (1-by-1 or 0-by-0) array types, and for the most part you end up working with array objects of type double. There's a lot of confusion here, because while everything can be an array in MATLAB, not everything should be. This makes MATLAB exceptional for use in applications where linear algebra is the primary use factor; however, it can be particularly painful when dealing with strings, or trying to implement specialized types you don't want to treat as an "array of something."

Python's type system, on the other hand, is much more rich than that of MATLAB's. Boolean values for true and false exist, for example, and distinct list / tuple / dictionary types are available, without conflating the ideas of one (arrays) into the other (structs). Unlike MATLAB, Python has true duck typing, allowing for heterogeneous data inside lists and dictionary types. A clear example of how MATLAB can be confusing where Python is not, is when dealing with arrays of multiple strings (e.g. a list of file names). In MATLAB, there is no direct string type, which forces you to use cell arrays instead of regular arrays to store several strings. Python, on the other hand (especially Python 3) has a very concrete and flexible string type, such that a list or array of strings does what you would expect, and even supports characters that are outside of the traditional ASCII set. For users outside of English-speaking areas, this is something MATLAB or C++ have begun to address in a user friendly fashion.

Interactivity and quick iteration through REPL

The REPL (read-evaluate-print-loop, or interpreter) is the main way to interact with Python and start writing code. REPL development is common in many dynamic languages today, and for good reason: Developing with quick, iterative feedback is a pleasant way to construct new models or solutions to complex problems. Being able to quickly visualize and work through a set of data is invaluable, and allows for a type of programming that provides quicker feedback, and encourages testing and curiousity over long sessions in the debugger.



```
In [1]: %pylab inline

        for _ in range(5):
            plot(cumsum(randn(100)))

        title("Some title")
        xlabel("X-axis label")
        ylabel("Y-axis label")

        Populating the interactive namespace from numpy and matplotlib

Out[1]: <matplotlib.text.Text at 0x7fdd484ca8d0>
```

An example of some code in a Jupyter notebook

If programming at the command-line isn't your cup of tea, the community has provided Jupyter, a package that allows one to work with what are called Notebooks. The idea of notebooks can be a little weird at first; however, there are a number of good reasons to use them. They provide a way to document and write code in the same place, and can even be set up to provide a web tutorial you can execute as you work through it. Jupyter notebooks are a great tool for collaboration, and make for what I believe to be a strong sense of community.

Standing on the shoulders of giants

Python's greatest strength, amongst everything else, is the expansive community it rests upon. Compared to many mainstream languages today, Python has access to third-party libraries for just about anything you could hope to use. Networking? Check. Web Frameworks? Check. Databases? Check. Statistics? Check. Numerical computing? Double check. Python is a great language to glue larger systems together, so you can collaborate better with others who are experts in their own fields. Even if you work entirely on your own, the vast majority of these packages are Open Source and are free for the community to use. Python now even comes with its own package installer, pip, just for this purpose.

The three main packages that are important for scientific computing, namely Numpy, Scipy, and Matplotlib, are available under permissive licenses and work with just about every version of Python out there. Numpy (http://numpy.org) provides multidimensional array and matrix types, as well as many linear algebra primitives, similar to much of the default functionality of MATLAB, out of the box. The Scipy project (https://scipy.org) is more interesting, and encompasses a much larger set of functionality, but supports many common statistical and mathematical operations that one might want.

The last package, Matplotlib (http://matplotlib.org), provides a very clean interface for producing graphs, plots, and charts in 2D or 3D reminiscent of MATLAB's plotting capabilities. Unlike MATLAB, however, Matplotlib supports unicode labels, anti-aliased fonts and lines in plots, and generally supports a wealth of customization options to produce professional, journal-quality plots that are descriptive and nice to look at.

The other strength of the community is in the breadth of support channels that exist for learning, teaching, and mastering the language. Companies from ESRI to Google, Facebook to Paypal have used Python in commercial projects deployed worldwide. Searching for articles on Python alone can be a daunting task; despite, or perhaps because of this, the maintainers and core members of the community are incredibly welcoming to new users of the language, and have provided materials for beginners to advanced users of the language alike to help produce beautiful and pragmatic code.

Modules and organization

Part of writing understandable and reusable code is organization. How does one organize code into sections, modules, or files? In Python, creating and organizing code is easy. Every file is treated as its own module, where the user can control which definitions in the file to export, which allows the user to control what interfaces are made available to the user and which aren't. This is quite different from MATLAB, which requires that only the first function definition in a file is exported, or even C++ where included header files pass on some of the clunkiness of early C programming.

Beyond the ability to reduce the file clutter within our projects, Python's module system also allows for control on what we import. This can be a huge lifesaver if you find yourself using separate libraries with similar functionalities. You might, for example, care that you are using the vectorized numpy.sqrt instead of the standard math.sqrt. While the benefits here may not be immediately clear, in the long run this can save some considerable headaches, as it can aid understanding of what your program depends on.

Utilizing the full power of modules allows for easy use and distribution of code. Often enough, it is as simple as dropping the Python source file into a relevant directory and importing the identifiers within it. Compared to MATLAB, which may require you to include several files or directories for a handful of functions, there is considerably less work when dealing with Python.

Where do I start?

If you're entirely new to Python, my first suggestion is to head over to https://python.org and download Python 3.5 (current as of writing). All of the major mathematical and scientific libraries support Python 3, and with the release of Python 3.5, the language has never been better than now.

If you are curious as to what kinds of libraries Python offers, or how to get them, I highly suggest checking the Python Package Index (PyPI, https://pypi.python.org). Python has everything from natural language tools (NLTK) to machine learning libraries (scikit-learn). The majority of these packages are quite well maintained, with continuous releases as often as every month or so. For Windows users who are having difficulties installing packages, Christopher Gohlke has provided a number of unofficial, pre-packaged packages for use at http://www.lfd.uci.edu/~gohlke/pythonlibs/. If you want to get up and running fast, these can be very helpful for trying out a package.

Final thoughts

As with anything, it is important to find the right balance when choosing how to approach a problem in research. When it involves something as complex as programming, this is even more true. C++ and MATLAB are two common languages that are taught to students who are new to Geomatics, and over the years they have helped solve many problems that researchers and engineers have faced. However, as our data grows larger, and our problems more complex, we need to adapt how we approach problems, so that we don't end up repeating work that's been done before, and so that we can focus on the solutions that are relevant to us, rather than struggling to get our tools to work for us.

Python provides a language, platform, and community to do this. Its dynamic typing and REPL provide a way to develop iteratively and rapidly. The quantity and quality of third-party libraries in the ecosystem make it possible to construct models for new problem domains, without first having to reinvent the wheel. Python has access to libraries from all walks of Geomatics, from databases and networking for GIS, to machine-learning and statistical models for image analysis and measurement. Most importantly, the community in Python is composed of experts from every field, and they readily provide resources for beginners. With Python, you don't have to be an expert in all types of programming to take advantage of the work the community has already done. Instead, you can build on that work, and stand on their shoulders. And since Python is Open Source, you can be sure that your code isn't going to stop working because you don't purchase another license.

Even so, you may not be convinced yet, but others are. ESRI has packaged Python to extend and interact with the ArcMap and ArcGIS environment, and others are following suit as well. Ultimately, people are realising that Python not only comes with a better community, but that it is more modular, maintainable, and most importantly, more expressive. Many of the things you might hope to achieve in Python cannot be done in other languages. This isn't just because of surface features such as syntax, but includes everything about the Python ecosystem from writing Hello World to deploying your code with Pythons package system. If you're at the end of your rope with the options you already have, or if you just want to try something new, I hope you give Python a chance. You just might find the key to success you were looking for.

Reference management with Zotero

By Ivan Detchev, University of Calgary, Canada

If you have ever had to write a substantial technical report, a research paper or a thesis, you most likely have had to deal with dozens if not hundreds of references. While you can manually handle citations and bibliographies for short assignments with relative ease, in order to be consistent longer ones require the use of reference management software. This article will briefly list common available programs for dealing with references, and it will introduce Zotero as a free alternative.

Obscured by formatting

My first exposure to reference management software was when I started graduate school. I had to write technical reports and research papers with at least half a dozen to a dozen references. The problem was that, depending on the organization the document was targeting, the styles for in-text citing and listing bibliographies differed. So I found myself repeatedly reformatting references I had already used. Reference attributes such as author name(s), date, journal name, volume, issue and page numbers were often listed in different order and surrounded by different punctuation. At this point I was introduced to EndNote – a program which allowed me to store all my references in a database, and reuse them in my papers with the appropriate citation and bibliography listing nearly automatically. The days of tedious reference formatting were over.

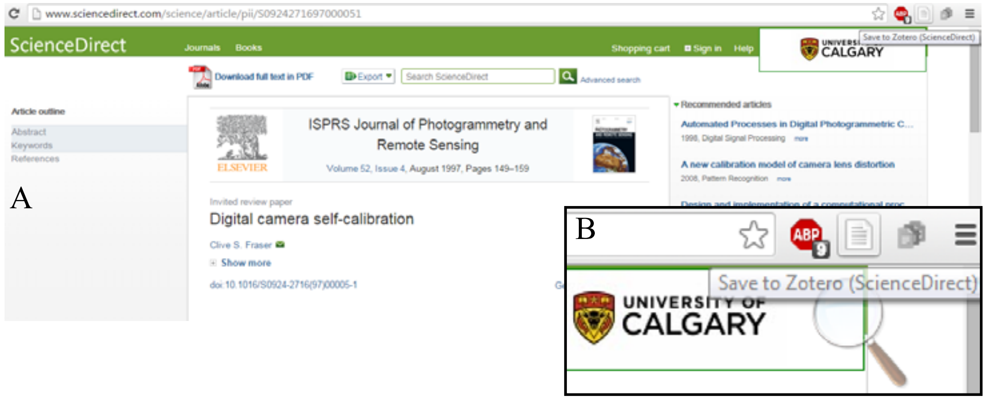
On the turning away

A few years down the road, my EndNote version was getting old and I was faced with the dilemma of either buying a new version or switching over to another program. Even though I had already accumulated close to 200 references and had several customized citation and bibliography styles, I decided to go for another program. The main options were RefWorks, Reference Manager, Mendeley, and Zotero. I didn't have the time to thoroughly explore all of them, but what caught my eye was that Zotero was free (and open source), it had a standalone version (so you could use a local database), it supported the main browsers (e.g., Firefox, Chrome) and document processing programs (MS Word, LibreOffice Writer), and it came with many referencing styles (e.g., ISPRS Journal of Photogrammetry and Remote Sensing, Journal of Surveying Engineering, Journal of Geodesy, IEEE). I was sold within an hour of using it.

Another reference in the database

Whether you need to transplant an already existing reference database or start a new one from scratch by adding new items individually, you won't have much trouble doing it with Zotero. The program supports several common reference management software file types, so both importing and exporting is possible. Adding new items to a database is also intuitive and effortless. All you need to do is click on the "Save to Zotero" icon on your browser (see Figure 1a and b), and the reference item is automatically stored on your computer along with its full text .pdf (if available) (see Figure 2). The items can be journal articles, conferences papers, books or other reference types. All the relevant fields for the specific type of item in question will be filled out, and they can be edited manually if necessary.

Alternatively, you may have the full text .pdf of an article, but not the citation information. In this case, you can add the file to Zotero, right click on it, and choose "Retrieve metadata from PDF". If the file does not have the digital object identifier (DOI) associated with the article embedded in it, choose "Create Parent Item" and manually key in as much of the citation information as you can gather from the document.



ISPRS Journal of Photogrammetry and Remote Sensing

Volume 52, Issue 4, August 1997, Pages 149–159

Invited review paper

Digital camera self-calibration

Clive S. Fraser

Show more

doi:10.1016/S0924-2716(97)00005-1

Automated Processes in Digital Photogrammetric C...
1998, Digital Signal Processing more

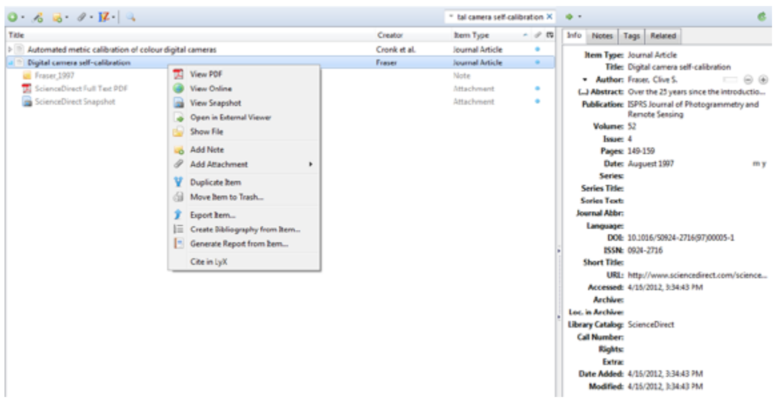
A new calibration model of camera lens distortion
2008, Pattern Recognition more

Design and implementation of a computational proc

Figure 1. Saving a reference item to Zotero from a web browser (a), and a zoomed in version of the "Save to Zotero" icon (b)

Learning to cite

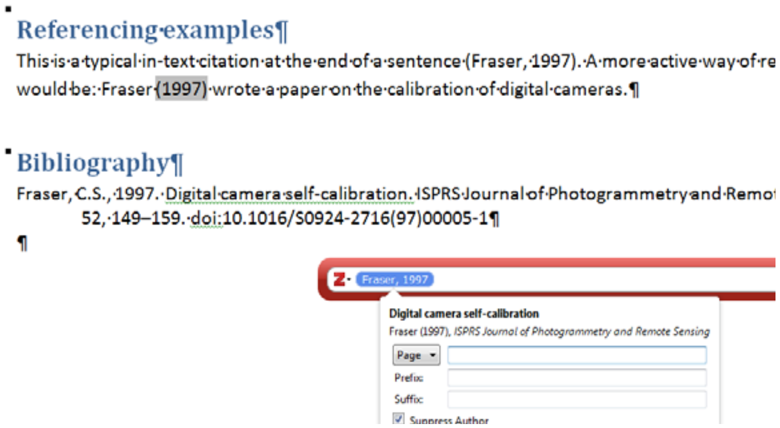
Once you have your reference database under control, you will be able to include in-text citations (a.k.a., "cite while you write"), and bibliography listings at the end of your document. I have been able to do this in both MS Word (using a Zotero add-in) (see Figure 3), and in LyX (using a plug-in called LyZ) (see Figure 4a and b). In MS Word, the add-in directly extracts the references from the Zotero database. In LyX, the plug-in first transfers a particular reference from Zotero to a BibTeX database, and then adds it to the actual document. LyZ also has the option to update the citations if the original Zotero database was modified. If you write directly in LaTeX, you can export specific items or simply your entire Zotero database to a BibTeX database, and then include that database in your .tex file. There are also tools for synching the Zotero and BibTeX databases.



Title	Creator	Item Type
Automated metric calibration of colour digital cameras	Cronk et al.	Journal Article
Digital camera self-calibration	Fraser	Journal Article
Fraser_1997		Note
ScienceDirect Full Text PDF		Attachment
ScienceDirect Snapshot		Attachment

Item Type:	Journal Article
Title:	Digital camera self-calibration
Author:	Fraser, Clive S.
(...) Abstract:	Over the 25 years since the introductio...
Publication:	ISPRS Journal of Photogrammetry and Remote Sensing
Volume:	52
Issue:	4
Pages:	149-159
Date:	August 1997
Series:	
Series Title:	
Series Text:	
Journal Abbr:	
Language:	
DOI:	10.1016/S0924-2716(97)00005-1
ISSN:	0924-2716
Short Title:	
URL:	http://www.sciencedirect.com/science...
Accessed:	4/16/2012, 3:34:43 PM
Archive:	
Loc. in Archive:	
Library Catalog:	ScienceDirect
Call Number:	
Rights:	
Extra:	
Date Added:	4/16/2012, 3:34:43 PM
Modified:	4/16/2012, 3:34:43 PM

Figure 2. Example of a journal article reference item



Referencing examples¶

This is a typical in-text citation at the end of a sentence (Fraser, 1997). A more active way of re
would be: Fraser (1997) wrote a paper on the calibration of digital cameras.¶

Bibliography¶

Fraser, C.S., 1997. Digital camera self-calibration. ISPRS Journal of Photogrammetry and Remo
52, 149–159. doi:10.1016/S0924-2716(97)00005-1¶

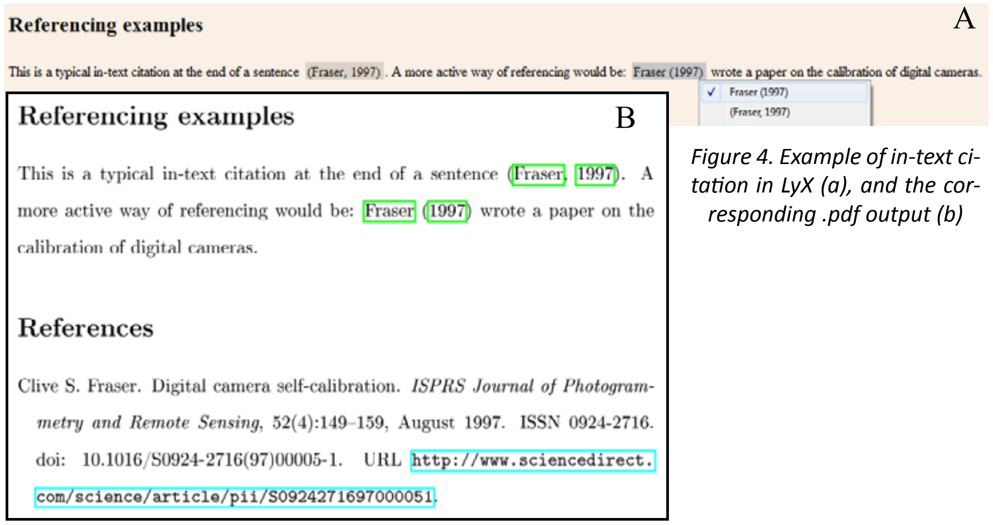
¶

Fraser, 1997

Digital camera self-calibration

Fraser (1997), *ISPRS Journal of Photogrammetry and Remote Sensing*

Figure 3. MS Word examples of in-text citations and bibliography listing



Referencing examples

This is a typical in-text citation at the end of a sentence (Fraser, 1997). A more active way of referencing would be: Fraser (1997) wrote a paper on the calibration of digital cameras.

Referencing examples

This is a typical in-text citation at the end of a sentence (Fraser, 1997). A more active way of referencing would be: Fraser (1997) wrote a paper on the calibration of digital cameras.

References

Clive S. Fraser. Digital camera self-calibration. *ISPRS Journal of Photogrammetry and Remote Sensing*, 52(4):149–159, August 1997. ISSN 0924-2716. doi: 10.1016/S0924-2716(97)00005-1. URL `http://www.sciencedirect.com/science/article/pii/S0924271697000051`.

Figure 4. Example of in-text citation in LyX (a), and the corresponding .pdf output (b)

The final cut

This article was intended to introduce the concept of reference management software and to present Zotero as a free alternative. Zotero is easy to install and set up. It is intuitive to use, so the learning curve is not intimidating. Once you build up a reference database, managing your references will be nearly automatic, so you will have more time to concentrate on writing the content of your papers.

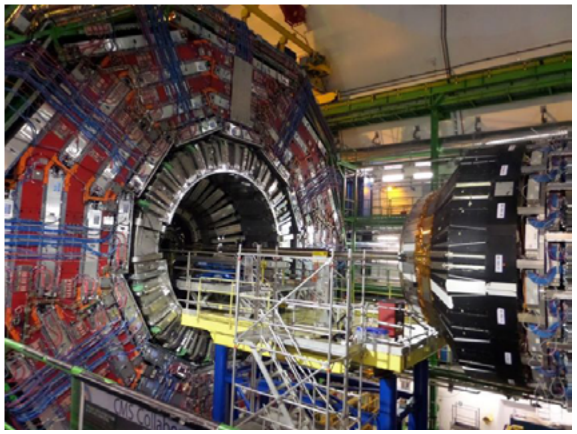
Help on the wing

If you have found this introduction to Zotero interesting, you can find more useful information and online tutorials on the following websites:
https://www.zotero.org/support/quick_start_guide
http://research.library.gsu.edu/zotero
http://libguides.mit.edu/zotero
https://www.zotero.org/styles
https://www.zotero.org/support/word_processor_integration
https://addons.mozilla.org/en-US/firefox/addon/lyz/

Zotero

The Annual Geomatics Engineering Trip to Switzerland

by Adam Clare, University of Calgary



For twelve days in May 2014, 25 students from the Geomatics Engineering department at the Schulich School of Engineering in Calgary, Alberta, were joined by their professor Dr. Michael Collins on a whirlwind tour through Switzerland. As a member of the Geomatics Engineering Student Society (GESS), I was responsible for organizing this trip. This was the eighth year of the trip. Every year this beautiful little country is visited by travelers from all around the world who come to take in its breathtaking landscapes, its unique cultural heritage, and its modern metropolitan centres. In addition to these attractions, Switzerland is also exceptionally interesting for students in the field of geomatics. This is because Switzerland has been for many years a center of excellence in land surveying, geodesy, cartography, photogrammetry, and many other geomatics disciplines. The unparalleled concentration of companies and institutions which make up the Swiss geomatics industry, along with their great hospitality toward our students, has kept us coming back year after year. The annual trip is still being taken in 2016.

Our first major destination was the United Nations (UN) in Geneva. The office is located in the Palais des Nations building, which originally housed the League of Nations. Many UN agencies are currently headquartered in Geneva, including the World Health Organization and the Office of the UN High Commissioner for Human Rights. Highlights of the tour included visits to the Human Rights and Alliance of Civilizations Room.

After visiting the UN, we proceeded to the European Organization for Nuclear Research (CERN), home of the famous Large Hadron Collider (LHC). At the time of our visit the LHC was under repair so we were able to go on an underground tour. The LHC is one of the greatest engineering accomplishments of the 21st century. It is 27 kilometres long, located 6 stories underground, goes through three countries and encompasses all disciplines of engineering. It was a huge privilege to be able to see it firsthand.

Swisstopo in Berne was the next visit on the trip. The Federal Office of Topography, better known as Swisstopo, is the government agency responsible for the measurement and geo-monitoring of Switzerland. The work of Swisstopo comprises the fields of geodesy, cartography, photogrammetry, remote sensing, surveying, and geographic information systems, all of which are core elements of the Geomatics Engineering program at the University of Calgary.

From Swisstopo, the Leica factory was next. Leica instruments are appreciated the world over by geomatics professionals for their quality and reliability. Virtually all of the levels and total stations used by students in surveying courses at the University of Calgary are manufactured either by Leica, or by its predecessor, Wild Heerbrugg. We were very fortunate to be granted access to their facilities for a fascinating behind-the-scenes look at what goes into producing the types of geodetic-quality instruments that are critical to the work of geomatics engineers.

Our final group visit of the trip was to the Swiss Federal Institute of Technology Zurich (ETH Zurich), the sister institute of EPFL. ETH Zurich is a very prestigious school for science, engineering, and mathematics, whose graduates and professors have to date been awarded 21 Nobel Prizes, including Albert Einstein in 1921. We were invited to visit the department of Civil, Environmental, and Geomatic Engineering at the institute's Hönggerberg Campus in the northwest of the city. There we were met by Professor Dr. Markus Rothacher, Director of Studies in Geomatic Engineering and Planning, along with several of his colleagues.

For the students on the trip, the acquired insight into the practices of our European counterparts will serve to broaden our perspectives as future engineers. This was truly an exceptional opportunity for all of the students to learn and develop, and the annual trip continues as a point of distinction for the Department of Geomatics Engineering.

Find us as : **ISPRS Student Consortium**

http://www.**linkedin**.com/groups/ISPRS-Student-Consortium-4510838

https://www.facebook.com/groups/isprssc/

Please visit our SC web page www.isprs-sc.org where you will find more information about Student Consortium, our previous Newsletter issues, SC activities, photo galleries from previous Summer Schools, interesting links etc.