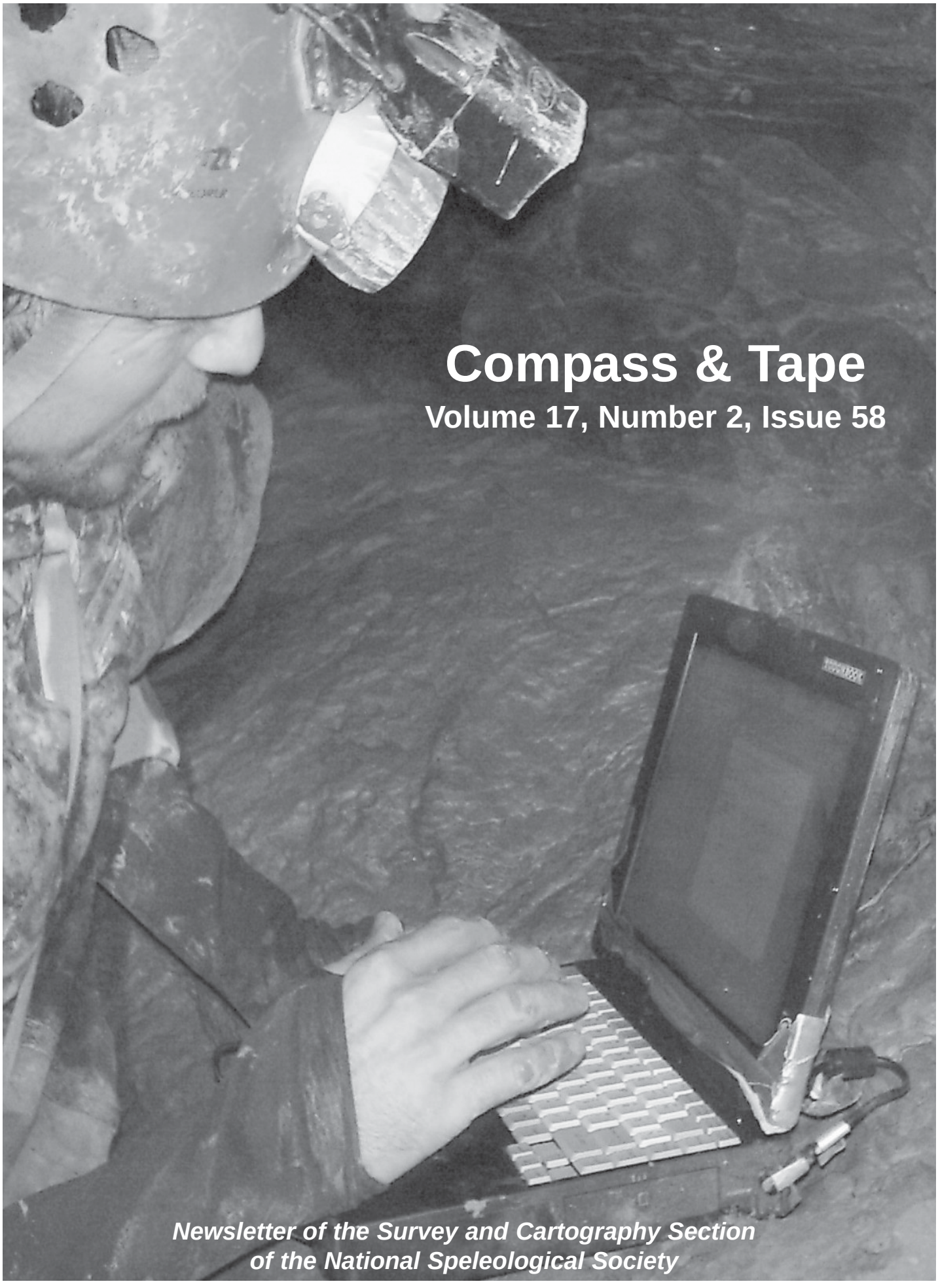


Compass & Tape

Volume 17, Number 2, Issue 58

Newsletter of the Survey and Cartography Section of the National Speleological Society

Survey and Cartography Section

The Survey and Cartography Section (SACS) is an internal organization of the NSS that is devoted to improving the state of cave documentation and survey, cave data archiving and management, and of all forms of cave cartography.
Membership: Membership in the Section is open to anyone who is interested in surveying and documenting caves, management and archiving of cave data and in all forms of cave cartography. Membership in the National Speleological Society is not required.

Dues: Does are $4.00 per year and includes four issue of *Compass & Tape*. Four issues of the section publication are scheduled to be published annually. However, if there are fewer, then all memberships will be extended to ensure that four issues are received. Dues can be paid in advance for up to 3 years ($12.00). Checks should be made payable to *"SACS"* and sent to the Treasuer.

Compass & Tape: This is the Section's quarterly publication and is mailed to all members. It is scheduled to be published on a quarterly basis, but if insufficient material is available for an issue, the quarterly schedule may not be met. *Compass & Tape* includes articles covering a wide range of topics, including equipment reviews, techniques, computer processing, mapping standards, artistic techniques, all forms of cave cartography and publications of interest and appropriate material reprinted from national and international publications. It is the primaly medium for conveying information and ideas within the U.S. cave mapping community. All members are strongly encouraged to contribute material and to comment on published material.. Items for publication should be submitted to the Editor.

NSS Convention Session: SACS sponsors a Survey and Cartography session at each NSS Convention. Papers are presented on a variety of topics of interest to the cave mapper and cartographer. Everyone is welcome and encouraged to present a paper at the convention. Contact the Vice Chair for additional information about presenting a paper.

Annual Section Meeting: The Section holds its only formal meeting each year at the NSS Convention. Section business, including election of officers, is done at the meeting.

Back Issues: SACS started in 1983 and copies of back issues of *Compass & Tape* are available. The cost is $1.00 each for 1-2 back issues, $0.75 each for 3-6 back issues and $.50 each for more than six back issues at a time. Back issues can be ordered from the Treasurer.

Overseas Members: SACS welcomes members from foreign countries. The rate for all foreign members is US$4.00 per year and SACS pays the cost of surface mailing of *Compass & Tape*. If you need air mail delivery, please inquire about rates. All checks MUST be payable in US$ and drawn on a U.S. bank.

Chair: Carol Vesely
817 Wildrose Avenue
Monrovia, CA 91016-3022
(818) 357-6927

Vice Chair Nigel Dyson Hudson
10 Slators Lane
Newfield, NY 14867
(607) 564-7927
ndh@cavesar.com

Treasurer: Bob Hoke
6304 Kaybro Street
Laurel, MD 20707
(301) 725-5877
bob@hoke.net

Secretary: Robin Barber
4312 Winding Way
Fort Worth, TX 76126
cavewoman@ev1.net

Editor: Patricia Kambesis
Hoffman Environmental Research Institue
Western Kentucky University
Bowling Green, KY 42721
ph: 270-745-5201
pnkambesis@juno.com

INSIDE

Front Cover: Mark Passerby uses Panasonic Toughbook CF-7 to enter data and sketch digitally in-cave. Photo by BVob Kirk

Back Cover: From the Zicafoose Cave Project PDF Map Book , cartographer: Mark Passerby

ISSN: 1074-596
Published in **November 2005**
by the Survey and Cartography Section of the
National Speleological Society.
Publishing Editor: Patricia Kambesis
Circulation Editor & Printing: Bob Hoke

The editor reserves the right to select which of the submitted materials will be used for publication. Of the material selected, the editor reserves the right to delete redundant or inappropriate material, to correct errors of spelling, grammer, or punctuation, and to edit for clarity, so long as such alternations do not change the meaning or intent of the author(s). In the event that significant changes are contemplated, the author(s) will be consulted and given the opportunity to review the changes prior to publication.

SUBMISSIONS

All types of materials related to cave survey and survey data, cartography, and cave documentation in general, are welcome for publication in *Compass & Tape*. Manuscripts are accepted in ANY form but are most welcome on CD's, 3.5 diskettes either IBM compatible or Mac format or via email. Typed material is next best although we will accept handwritten material as long as it is legible. Artwork is any form. shape or size is also welcome. Send all submission for *Compass & Tape* to:

Patricia Kambesis
Hoffman Environmental Research Institute
Dept. of Geography/Geology -
Western Kentucky University
Bowling Green, KY 42121
270-745-5201
Email: pnkambesis@juno.com

The Raders Valley Project

by Mark Passerby

Raders Valley, located just west of Lewisburg, WV, has long been our diggers paradise. Most of its known caves have been discovered over the years by our group's digging efforts. Zicafoose Blowhole, Bobcat Blowhole, Freelanders Well, Middle Earth, and Deels Hole are a few of the substantial caves that now tally over 4.5 miles of previously unknown passageways. Exploration and mapping of these caves have begun to help unravel the mysteries of the many blowing holes that continue to serve as our "diggers motivation."

My personal involvement in the project began 13 years ago with the successful opening of Zicafoose Blowhole. The cave quickly became exceptionally tough and proved to be beyond my comfort level at the time. Fortunately, on the night of the initial breakthrough Jim Tompkins and Mike Dore had become involved. Jim, Mike and a few others —largely motivated by the energy of Jim Tompkins — would go on to discover about a mile of passage that is now displayed on page 1 of our 35-page, PDF-formatted working map. (View Map Book online at www.caves.com/zicmap.pdf). But the source of the wind could not be found and exploration in Zicafoose Blowhole ceased.

Some 13 years later Bob Kirk, Aaron Bird and I would return to Zicafoose Blowhole and enlarge the tight areas, step by step, using a clever invention of Bob's that we dubbed "muzzminers." These rock rippers paved the way for discovery! The new discoveries and subsequent mapping in Zicafoose, as well as Bobcat, Middle Earth, and
Deels Hole, created a pile of survey data that would become the evolving focus of my interest.

Choosing Software for the Project

The project initially used Compass Survey Software for its data processing. Data was compiled and a line/wall plot generated. Then, using various other programs I would "morph" the scanned sketches to fit the line plot. This approach ultimately resulted in a working map of decent quality, albeit still not much closer to a final drafted map. Other programs, such as Carto and Winkarst, have continued or expanded on this raster/scanned sketchbook approach, with Carto even beginning to add some basic drawing functions. These raster approaches all share a common foundation of assembling sketches and working map pieces, then virtually tacking them to the line plot. If the survey changes due to corrections and loop closure, the composition of scanned and assembled sketches will adjust accordingly.

Author using Fujitsu Stylistic 2300 in Zicafoose to enter data and add detail to running lineplot.

The scanned sketch approach, however, doesn't force the project cartographer to actually begin drawing or working towards a final high-quality map. The lack of a "draw as you go" workflow has often produced (as many of us can recall) huge piles of accumulated data and stacks of sketch books full of long since forgotten passages. This kind of project eventually requires volunteers to devote large amounts of time to attempt a final map draft. And this is often made more difficult due to missing sections of survey, inaccurate data, and poor sketch quality. Thus the term "remap" was born.

A "draw as you go" workflow spreads the drawing workload over the life of the project and additionally provides tangible "near finished" results to those involved in the project. Ultimately, this jump forward provides exponential benefits and brings into the mix a dizzying array of new ways to present and explore the visual data. But at the same time, the digital working map must correctly conform to an ever-changing survey database.

Vitally important to this process is software that treats various objects of the working map differently when features need to be adjusted. Passage outlines, for instance, must be reshaped or "morphed" in a way that depends on the shifts of nearby survey vectors. Other features, such as text, symbols, and cross sections, must be translated and scaled so that they keep their survey-relative positions. If this is not done regularly, the working map can't be used directly to produce an accurate finished map or in some cases even a useful provisional map. The drawings simply won't align with the latest data compilation and resultant line plot.

Until now this has been the cave cartographer's main problem with the approach of "draw as you go," especially as it relates to large cave systems where many line plot shifts and data changes occur through error discovery
and loop closure.

The SVG Solution

The solution, which I will discuss in brief here, is a quickly emerging image format named Scalable Vector Graphics (SVG). This complex 2D format was spurred forward in 1998 by proposals submitted to the World Wide Web Consortium (W3C) by software developers, including Adobe, IBM, Netscape, Sun, HP, Macromedia, Microsoft, and Visio. This led to the first draft of the SVG Specification in February of 1999. From this beginning point the SVG format, which is XML-based, quickly took hold among cartographers and GIS specialists. How the format benefits the cave cartographer is nothing short of astounding!

Walls Survey Software and the Raders Project

During the 2003 NSS Convention in California I had the good fortune to attend a talk by George Veni who demonstrated a program called Walls. Quite ironically, at the same event I was giving talks on morphing working sketch scans or what are more commonly known as raster images. In contrast to this approach, Walls deals directly with vector-based images and more importantly with the problem of "merging" them with output from a commercial drawing program. Veni's demonstration, as well as the demo in the Walls program download, shows a cave with hundreds of loops and drawn wall outlines being morphed to fit a very complex adjusted line plot — in seconds! Absolutely amazing, is what I thought as I walked out of the room, dizzy with thoughts of new possibilities for the Raders Project. Now I would actually have to draw, and not only draw but also become proficient enough to keep the drawing updated with the continuing in flux of survey data and sketches.

Zicafoose Blowhole would become the first cave in the Raders Project to have a "draw as you go" map that automatically tracks an ever-changing set of survey lines. Adobe Illustrator (v10 or CS) was chosen as the drawing tool because of its ability to import and export SVGs that become part of a process known as roundtripping. Roundtripping, however, has no utility in small caves, or for that matter caves whose surveys contain no loops. It is strictly a function for bigger cave projects where error discovery and loop closures require readjustment of the line plot. For such projects roundtripping becomes an essential part of the "draw as you go" method of mapping.

SVG Roundtripping

The process begins with raw survey data entry in Walls. The data is then compiled and exported as a line plot in the SVG file format. The file's content is actually more complex than a simple line plot as it contains predefined layers and special tags that make subsequent versions of the map updateable. Next, the file is opened or placed in the master Illustrator document, the "draw as you go" cave project, where complex detail (such as boulders, formations, and wall outlines) is added and placed in the pre-defined layers as described in the Walls manual. Once the detail is added, the document is saved from Illustrator as an SVG image. Usually "_mrg" is appended to the file name to signify to the cartographer that the file is usable as a "merge" file, or source SVG for Walls.

After the next survey trip more data is entered into Walls and another SVG image generated. This time though, in the Walls SVG Export dialog, the latest SVG file saved from Illustrator is selected to be an SVG source file. Whenever such a file is specified, the export operation will adjust and merge its contents with the latest set of survey data objects. After a mouse click and a few seconds' wait, an updated SVG map is produced.

Once again this new SVG file from Walls is opened in Illustrator so that more detail can be added

to the growing cave's line plot. The roundtripped SVG is therefore sharing additions from both Walls and Illustrator, with Walls adjusting Illustrator artwork to fit an evolving cave survey. From here, the sky is the limit.

Off and Running

After several trips the processes necessary to maintain a correct "draw as you go" project became clear, but something else became evident to me. Buried within the Walls/Illustrator roundtripping process is the ability, while in the cave, to rapidly add incredible amounts of detail "on the fly" to a running survey line plot. This would require the use of a ruggedized laptop or tablet PC. Getac (www.getac.com) provided us with our first opportunity to put my ideas to the test on the Raders Project. They sent us their Model W130, running Windows XP, with a touch screen and full keyboard to use in-cave for a couple of months. For transportation we used (and still use) modified Pelican 1490 cases. These stout cases can easily carry the rugged PC and an external battery pack for extended run times, with room for some personal gear as well. If need be, the remainder of personal gear normally carried by the sketcher on a traditional cave survey can be divided over the remaining team members.

The process does indeed work. As my knowledge of what it takes to draw electronically in-cave has increased, several critical factors necessary to realize real gains in efficiency have become evident. First, the in-cave electronic sketcher needs a high level of out-of-cave experience with the Illustrator program. This familiarity combined with a master Illustrator file that's fully stocked with an organized library of drag-and-drop cave symbols and brush patterns can make the process very rewarding and efficient.

Though labor heavy on the font end, such preparation can result in rapid gains in efficiency once you're fully set up! A pile of breakdown becomes as simple as tap, stroke, and then move on to the next item of detail. A soda straw: just tap, tap, and move on. After the trip, the detailed drawings done electronically in-cave are added to the master "draw as you go" project file. Then the fun really starts!

A Note About Ruggedized PC's

Obviously a $4,000-plus Getac is out of reach for most of us, so I began to experiment with other

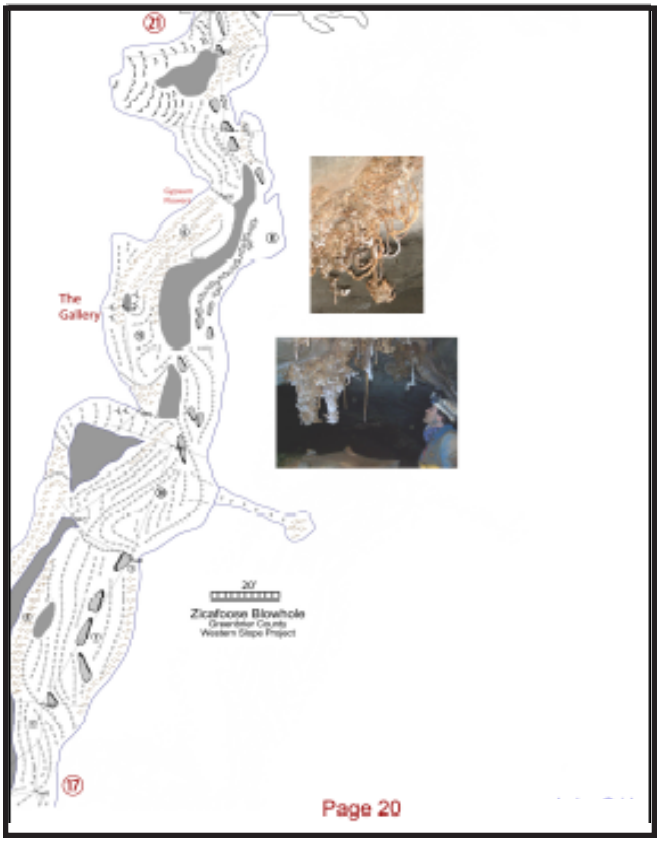
Bob Kirk carrying the modified Pelican 1490 into Zicafoose Blowhole for an in-cave draw as we go survey trip

units as I became familiar with the processing needs of the "draw as you go" project. For Zicafoose Blowhole we currently have a Fujitsu Stylistic 2300 tablet PC running Win98 with 160 MB of RAM and a 266 MHz processor. This processor/RAM configuration is fine; however, the Stylistic tablet models lack keyboards for quick data entry and are not ruggedized. Therefore, both Bob and I have purchased Panasonic Toughbooks (Model CF-27) with 266 MHz processors and 160 MB of RAM running Windows 98. These two units, which we found at less than $200 each, have a keyboard and are exceptionally rugged.

Additionally, the Toughbooks are compatible with our choice of external batteries needed for the extended run times of a cave survey. NOTE: The current version of Illustrator (CS or v11) won't install on Windows 98 but Illustrator 10 is compatible and still available online on sites such as Ebay.

The PDF Map Book

Perhaps the most rewarding part of the entire Raders Project has become what I have dubbed the PDF Map Book. The electronic format of this book (PDF format) allows the project participants to completely explore the cave project in incredible detail, page by page, via simple mouse clicks. Online it will display in the project participant's web browser in a lower resolution called "Fast Web View." Alternatively,



From the Zicafoose Cave Project PDF Map Book

View Map Book online at www.caves.com/zicmap.pdf

Cartographer: Mark Passerby

it can be saved to a PC/MAC desktop and opened in Adobe's Acrobat Reader with a much higher display resolution. Each page or the entire book can be printed for hard copy reference as well. The Raders Project participants will receive, generally within a week of a survey trip, an updated link to the newest version of the PDF Map Book where they can view and explore the fruits of their labor. (You may view the PDF Map Book for Zicafoose Blowhole by going to www.caves.com/zicmap.pdf. The aerial and topographic overlay pages are disabled to protect the cave's entrance location from general online display).

From the cartographer's point of view the map book is once again labor heavy on the front end. After formatting is complete, however, the addition of new working map pages becomes quite simple. Three products are used regularly in the PDF Map Book portion of the project: Adobe Illustrator, Adobe Photoshop, and finally Adobe Acrobat to generate the PDF Map Book file.

Complexities Made Simple

The PDF and SVG file formats both give the digital cave cartographer, through carefully selected choices, the ability to display incredibly complex sets of overlapping passages. Within SVG this is accomplished by subgrouping levels of passages nested inside the pre-defined groups of the original SVG generated by Walls.

The SVG can then very easily be embedded in an HTML Web page. Placing simple scripts inside the HTML portion allows certain functions (such as turning selected layers on and off) to be performed by the online user. For example, if three levels of passages were stacked upon each other, it would be easy to script a few check boxes to allow the user to view the different levels independently. An early map of Zicafoose Blowhole that has this feature is accessible at www.cavediggers.com/fullmap2.html. (Viewing it on

a PC requires Internet Explorer with Adobe's SVG plug-in installed.)

I suspect that with the growing popularity of SVG, images in that format will become natively viewable in most browsers without the need to download the plug-in from Adobe's Web site. The PDF version allows for complex overlapping passages to be displayed, with the cartographer choosing colors for passage floors or outlines to signify different levels. The levels can be made "clickable" within the PDF book so that a user wishing to view only a particular level can, with a mouse click, jump to a corresponding set of map book pages. Obviously, when the cave system is very complex the decisions of the project cartographer will bear heavily on the overall functionality and ease of use of the PDF Map Book.

Details and Purchasing Information for Items Mentioned

1) Starting set of Illustrator Cave Symbols at *www.cavediggers.com/Symbols.zip.*

2) Walls Online Printable PDF Format Manual at *www.cavediggers.com/walls32.pdf.*

3) Walls Survey Software at *www.utexas.edu/depts/tnhc/.www/tss/Walls/tsswalls.htm.*

4) External batteries (15v) to run the Toughbook line of rugged PCs are approximately $180 each with shipping. (See model PM-148 at www.bixnet.com/unpowbat.html.)

5) Pelican 1490's can be found on Ebay for $125-140 and modified with hardware found locally.

6) For Toughbooks (CF-27) and other new or refurbished other models, contact Greg Doyon at Telrepco, Inc. (203) 284-5239, (800) 537-0509 x239,gdoyon@telrepco.com , *www.telrepcoPCstore.com.*

Protecting Your Suuntos

by Marc Ohms

The latest models of Suunto's compass and clinometer have a much thinner plastic capsule than previous models. At Wind Cave National Park we have had eight instruments break within a year and I have heard from many others that they have had this same thing happen. The plastic is very thin and cracks very easily. I wrote the company but all I heard back was the typical "*thank you for your concern…*" Getting nowhere with the company I decided to strike out on my own and devise a way to protect the instruments.

On my first attempt I purchased plexiglass in a thickness of .1" and had the store (Lowe's) cut it into 2 inch squares. I then used a silicone epoxy to attach the square to the top of the instrument. The square fit perfectly except for the two upper corners which hung over the edge a bit. I simply used a file to round the edges flush with the instrument. On my second attempt I used a drill with a 2 inch hole bit to cut circles, which fit perfectly and did not require any filing.

After allowing the epoxy to dry overnight I moved on to the next step, which Mike Yocum came up with in 1995. I purchased Plasti- Dip (*there are several brands of this and go by several names*), which is the stuff you cover tool handles and such with. The container for the Plasti-Dip is just wide enough to allow you to dunk an instrument. Before dunking into the dip, I thoroughly cleaned the instrument. Next I stuck a cork into the eyepiece so the dip will not run into it. I then dunked the instrument into the dip then hung to dry. Depending on the temperature and humidity this may take minutes to hours. I repeated two more times for a total of three dunkings and allowed this to dry overnight. When it was dry I took an Exacto knife and cut around the cork, then removed the cork. Although they make the dip in clear, I could not find it. If you use the clear, then you are done at this point, as enough light should pass through to illuminate the instrument. If you are like me and could not find the clear (*or simply want pretty colored instruments)* then you will need to cut a piece of the coating from the window. I used a spool of thread to trace around with the Exacto knife and removed the middle. Presto! I just dramatically extended the life of my instruments. They are as bombproof as they can be. We used two different colors for different instruments. We used red for compasses and yellow for clinometers. Even in the cave you can easily determine which instrument is which.

The dip protects from the bangs, knocks, and clanging they receive while being drug through the cave and used during a survey. You can buy the prefab plastic sleeves or for the same price buy enough dip to coat 5 sets of instruments. Plus the dip provides additional protection from water and dirt that the sleeves do not. I have only dipped the KB-14 model and have not attempted the tandem or models with the focusable eye piece, but I am sure both are doable with a little ingenuity.

Protecting your instruments from damage and destruction not only protects your investment but protects your survey accuracy as well. Every bang and knock your instrument receives can be damaging and result in errors in your survey. Even if you do not see exterior damage the insides can be getting damaged.

Yocum, Mike. 1995. PROTECTING SUUNTO INSTRUMENTS. Compass & Tape, v. 12, no. 1, p. 9-11.

Finding Loops In Cave Surveys

by Larry Fish

Most cavers understand the concept of survey loops. On the surface, it appears to be a pretty simple idea: you have a loop in a cave survey when a series of shots circle back to the same starting point. However, in spite of this simplicity, finding loops with a computer program is quite difficult. In this article, I will discuss the problem of finding loops with computer programs and talk about solutions to the problem.

Simple Loops.

If you are not picky about which loops you choose, finding a loop is pretty easy. The problem becomes difficult if you want to find the "simplest" set of loops in a cave. For example, look at the following drawing:

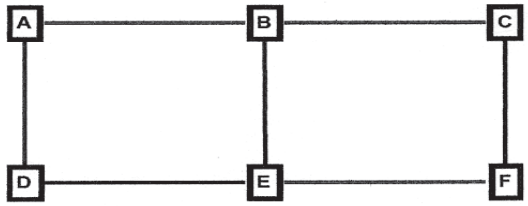


Figure 1

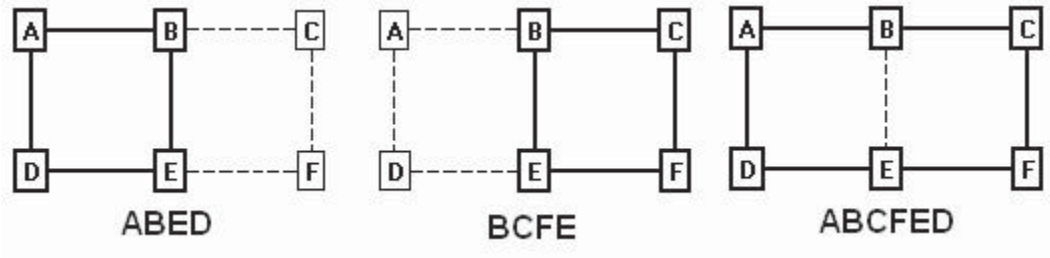
Here we have a simple pattern of shots that produces three possible loops. The first loop consists of the stations A, B, E, and D. The second consists of B, C, F, and E. The final loop consists of stations A, B, C, D, E, and F (Figure 1A).

Now the question becomes, which of these loops are most useful to cave surveyors? To satisfy the needs of surveyors, the set of loops that we choose must satisfy three requirements:

1. All Shots. The loops we choose must cover all the shots in the cave that are included in a loop. In this example, every shot is part of one or more loops, so we must include every shot. No one loop covers all the shots but if we choose any two of the loops, all shots will be included. That means one of the three loops is superfluous and can be ignored.

2. Smallest Size. The loops we choose should be as "small" as possible. Depending on your needs, "small" could mean the fewest number of shots, the smallest perimeter or the smallest enclosed area. For cave surveyors, choosing loops with the smallest number of shots is probably the best choice. This is because having fewer measurements makes it easier to find blunders. Also, if you have to repair a bad loop in the cave, it is a lot easier to resurvey a small number of shots.

Figure 1A



3. Minimum Overlap. The loops we choose should also have the smallest possible overlap. The reason for this principle is obvious. The less overlap between loops, the easier it is to isolate and locate blunders. Also, if you are closing loops by distributing errors around a loop, having loops with minimal overlap prevents good loops from being distorted by bad loops.

Going back to Figure 1 above, to satisfy the requirements, we would choose the loops (ABEF) and (BCDE), and not (ABCDEF). To clarify why this combination is preferable, look at the follow chart:

Loop1	Loop2	Total Length	Overlapped Edges
ABEF	BCDE	8	1
ABEF	ABCDEF	10	3
BCDE	ABCDEF	10	3

The chart shows how various pairs of loops compare. As you can see, the combination of the two smaller loops produces the shortest total length and the smallest number of overlapped edges. This pair of loops would be preferable for cave surveyors.

Finding The Smallest Loops.

With the simple example, it is fairly easy to pick out the smallest loops. However, as you add more and more interconnected loops, the problem becomes complex, even for a computer program. It turns out that the problem is "NP-Complete," [1, 2] which is a mathematical term that means you have to try every possible combination of loops before you can be sure that you have the best set of loops. Since the number of possible loops goes up exponentially with the density of the edges, the time to solve the problem can be exponential. This could make the problem too time-consuming to be practical for cave survey programs. For example, it might take a few seconds for small cave, hours for a medium sized cave and months for larger caves.

Dealing With NP-Complete Problems.

It is often possible to deal with NP-Complete problems by finding a faster solution that does not give mathematically perfect results. In the case of cave surveys, after several months of experimenting, I was able to find a practical solution that gives nearly perfect results and runs relatively fast. To save other surveyors and programmers some time, I will outline the method I used in detail.

Graph Theory.

The branch of mathematics that deals cave surveys and loops is called "Graph Theory." [7] Graph theory is pretty easy to understand as long as you know the language. For example, in Graph Theory, the stations and shots in a cave survey would be called a "graph." A "Station" is called a "Vertex" and a "Shot" is called an "Edge." A Loop is called a "Cycle," and when you find all the Cycles in a Graph it is called the "Cycle Basis" for that graph. Here is a chart that translates the language of cave surveying to graph theory:

Cave Survey	Graph Theory
Station	Vertex
Shot	Edge
Survey	Graph
Loop	Cycle
All Loops	Cycle Basis
Smallest Loops	Minimum Cycle Basis

Spanning Trees. The first step toward finding loops in a graph or a cave survey is creating something called a "Spanning Tree." In a cave survey language, a Spanning Tree would be a series of shots that touch each station in a cave only once. In other words, every station has no more than one

shot leading to it. For example, here is a Spanning Tree for Figure 1 above:

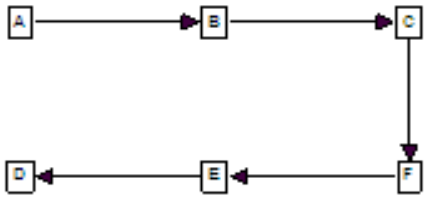


Figure 2

As you can see, the Spanning Tree touches every station in the cave without visiting any station twice. The shots A-D and B-E have been removed from the survey because, otherwise, stations D and E would have two shots leading into them.

Multiple Spanning Trees. Another important aspect of spanning trees is that every graph can have more than one valid spanning tree. For example, here in another spanning tree for Figure 1:

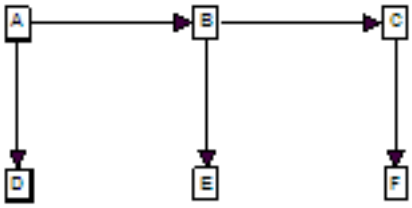


Figure 3

As you can see, this spanning tree also has one shot leading into each station but it uses a different set of shots to do it.

Finding Loops In Spanning Trees. Another way to think about this idea is to realize that a Spanning Trees is just a cave survey where one shot has been removed from each loop, effectively breaking that loop. That means that all you have to find a loop in a Spanning Tree is to put one of the "missing shots' back. For example, in the Figure 3 above, if you put the shot "D – E" back, you will have found the "ABED" loop. In other words, for every shot that is missing from the Spanning Tree there is one loop. Thus the "missing shot" is the key to each loop and we will use this shot as a kind of marker for the loops in a cave.

To actually figure out which shots belong to the loop, you follow the "parents" of the missing shots back until they come to a common station. (The arrows in the example graphs show which stations are the parents.) For example, in Figure 3, you follow E back to B and B back to A. This defines one side of the loop. On the other side, you follow D back to A. You stop at A because it is the common parent of both sides loop.

Problems With Spanning Trees. In the language of Graph Theory, the loops generated using this approach are called "Fundamental Cycles." Unfortunately, Fundamental Cycles do not satisfy the criteria that we have set out for cave surveying. In graph terminology, what we are looking for is called the "Minimal Cycle Basis" of a graph. Although the previous example seems to work perfectly and does find the Minimal Cycle Basis, this is not always the case. For example, here is another example of a spanning tree:

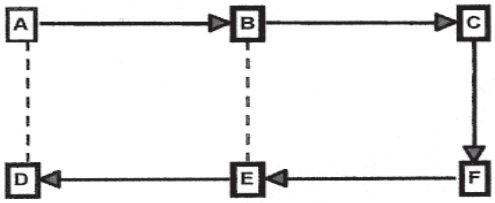


Figure 4

If you look at the loop that is formed when shot A-D is put back into the spanning tree, you will see it consists of stations A, B, C, D, E, and F. This is the one loop from Figure 1A that we decided was defective.

Different Kinds Of Spanning Trees. You can now see that some kinds of Spanning Trees work better than other kinds. As a result, one way to improve our loop-finding ability is to use different kinds of Spanning Trees.

Finding Spanning Trees. To understand how to improve Spanning Trees, it is important to understand how they are derived. The process is pretty simple and similar to the way we explore a cave. Generally speaking, you simply "explore" the whole cave and put every shot you find into the spanning tree unless you have seen the station before. The key to this process is how you "explore." There are two basic methods of "exploring": "Depth First Search" (DFS) and "Breadth First Search" (BFS).

Depth First Search. The Depth First Search explores by going to the "back" of the cave first and then working your way back forward. In other words, you do not linger at each station and you do not initially explore all branches. Instead, at each station, you pick the first passage you see and follow it deeper. You don't back track until you reach the end of the cave. Here is an example of the a DFS spanning tree for Figure 1:

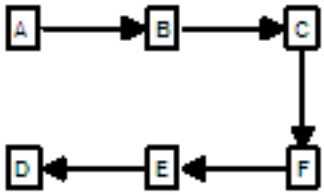


Figure 5

In this example we start at A and keep going deeper until we run out of new shots at D. Then we backtrack to E and look for any side branches. There is a side branch running from E to B, but we have already visited B, so we ignore it. This process of backtracking continues until we have checked all side branches and worked our way back to A.

You will notice that this gives the same spanning tree that we had in Figure 4, which produced the bad, overlapping loops. For the purpose of finding minimum cycles, a Depth First Search produces poorer loops.

Breadth First Search. With the Breadth First Search, we explore the front of the cave first. We do this by checking all the branches off each station, before we move any deeper into the cave. For example, here is a Breadth First Search spanning tree for Figure 1.

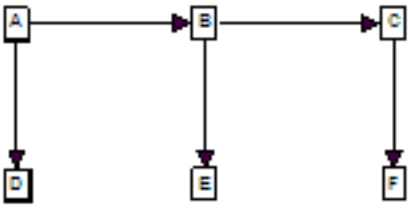


Figure 6

In this example, we start at station A as before, but we explore the connections to B and C before moving on. When we do move on to B, we explore all its connections too and continue until we run out of stations that have not been visited. You will notice that this gives the same spanning tree as Figure 3, which we concluded, gives superior loops. So for our purposes, BSF spanning trees give superior results.

BSF Problems. Unfortunately, as the loops get more complicated, the BSF begins to have problems. For example, here is a more complex survey and the BSF spanning tree it produces:

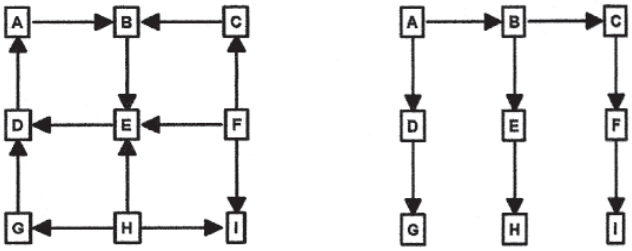


Figure 7

As you can see, this spanning tree will produce two sets of overlapping loops, which is just what we are trying to avoid. One way to avoid this problem is to change the way we explore. For example, if we preferentially explore the stations

with the most connections to them, it will improve the spanning tree. In the example above, we would choose E first, because it has four shots connected to it. Next, we would explore B, D, F and H because they have three connections. This is the result:

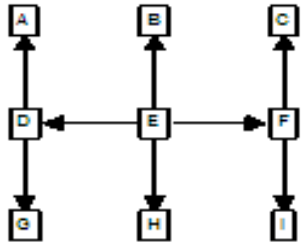


Figure 8

As you can see, this produces the kind of minimally overlapped loops that we are looking for. (Note: detailed information on implementing these BFS algorithms is available in Manuel Huber's article [3] in bibliography.)

The Final Problem. Unfortunately, improving the spanning tree can only be carried so far. Eventually, if you have enough loops, even an optimized spanning tree fails to produce a Minimal Cycle Basis:

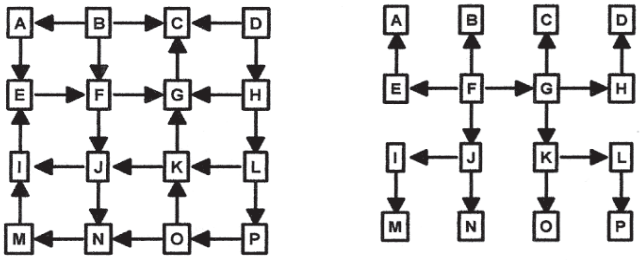


Figure 9

In this example, we have a 3x3 "grid" of loops. As you can see, even using the optimized spanning tree algorithm, two of the nine loops in the area of F, G, J, K, N and O will be overlapped. Thus, when you have loops stacked three-deep in two directions, even an optimized spanning tree will generate overlapped loops.

A New Strategy. At this point, we are forced to adopt a new strategy for improving loop quality. We know that a modified BSF spanning tree will find a set of loops that almost satisfies our criteria, so all we have to do is fix those loops that are sub-optimal. The basic idea is to find a method of finding and fixing sub-optimal loops. During my experiments, I found several methods that could be used to find bad loops and fix them. The fastest method was to look for "short cuts".

Looking For Short Cuts. To find short cuts, you look at each loop to see if there is a shorter alternate route. The process begins by selecting one station from the "missing shots" and then doing a Breadth First Search of all adjacent stations. If during the course of the BFS, you find the other end of the "missing shot," you have a short cut. Since the BFS only explores one "level" of depth at a time, the depth of the search tells you the length of any potential short cut. If the depth exceeds the length of the loop your are trying to improve, you know there is no short cut.

The process is a bit complicated, so here are step-by-step instructions:

1. Choose one loop and note the length of the loop.
2. Choose one station from the "missing shot" for the loop.
3. Do a Breadth First Search of all adjacent shots to this station.
4. Each time you move deeper in the search, keep track of the parent shot.
5. Each time you move deeper, keep track of the depth.
6. If the depth is greater than the length of the loop, there is no short cut for the loop, so stop searching.
7. If you encounter the other station in the "missing shot," you have found a short cut.

8. Create a new loop by collecting all the shots in the short cut. This is done by following the parent shots back to the starting station.
9. Mark the "missing shot" so it is excluded from all futures short-cut searches.
10. Repeat steps 1 through 9 until all loops have been checked for short cuts. You can ignore any loop with less than four shots.

Note: this algorithm is described in detail in Manuel Huber's article [3] in bibliography.

Loop Order. Each time you find a short cut, you eliminate the "missing shot" from the search. This assures that you do not find any overlapping short cuts. However, it can cause the program to miss certain other valid short cuts. For this reason, the order in which loops are processed is important. There are several different ways to order the loops for processing, but for a variety of reasons, I chose to sort all the loops by length and start with the longest loops first. [3]

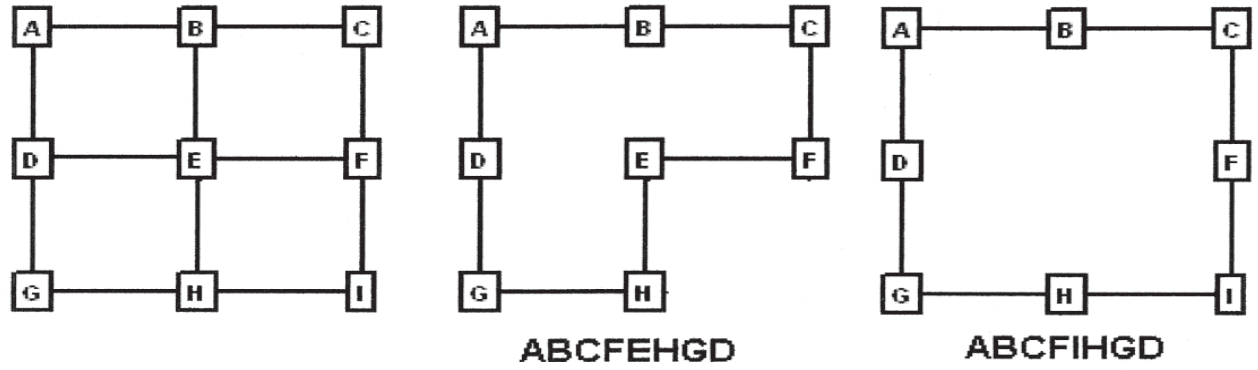
Problems With Short Cuts. The "Short Cut" technique works properly for 99% of the loops. However, there are still a few cases where the technique finds sub-optimal loops. This is typical of an NP-Complete problem and we would normally be satisfied with a few sub-optimal loops if the algorithm were fast. In this case, I found a way to improve the technique so it generates nearly 100% minimal loops. (At least in my testing of thousands of random loops, I have yet to find a failure.)

The "Short Cut" technique fails in certain situations where you have two overlapping loops that have the same number of shots. Here is an example:

On the left, we have a pattern of shots that forms four minimal loops and nine additional, overlapped loops. In some instances, the Spanning Tree will find the two overlapping loops shown on the right. (This depends on the order of the shots and how the loops are connected to the rest of the survey.) Since both loops have the same length, the program may choose either loop to process first when it starts to look for short cuts. If the program chooses to process the middle loop first, shots H-E or E-F may be prematurely eliminated from consideration as a short cut. This may cause the program to select overlapped loops around stations {A,B,C,D,E,F}.

Enclosed Area. My solution was to sort the loops by area whenever I found two loops of equal length. This was a bit more complicated to do than it sounds. While it is easy to find the area of an irregular 2D polygon using the "Surveyor's

Figure 10



Rule" or "Determinants", the surface area of a 3D polygon is difficult to define and calculate. As a result, I chose to find the center of the polygon (centroid), and calculate the average distance to each vertex. This is relatively fast and easy to calculate and gives good results in testing.

Minor Issues. When you combine all the steps outlined above, you have an algorithm that finds minimal loops nearly 100% of the time. In spite of this, there are some situations where the algorithm will find loops that "appear" to be sub-optimal. For example, look at the following diagrams:

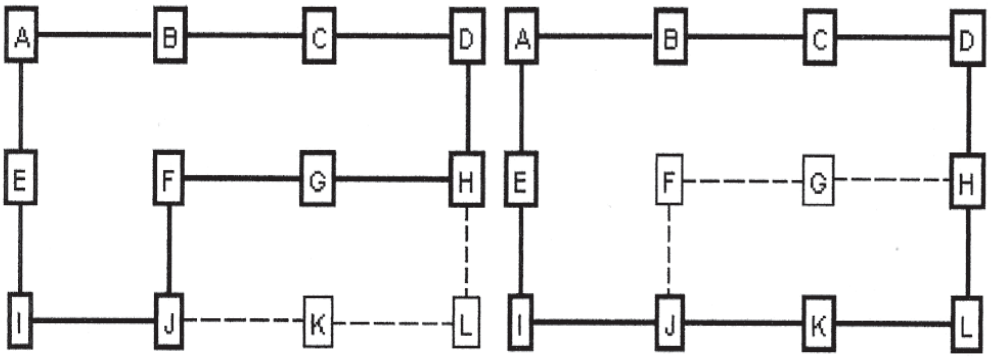


Figure 11

Here you have two possible loops in a graph. Both loops satisfy the requirements of having the minimum number of shots and overlapped edges. However, most people would probably choose the loop on the left as the smallest loop because it has a smaller area. For the purpose of cave surveying, there is no advantage to either loop and both would work just fine. You could add an extra step to the algorithm to fix these cosmetic defects, but it is probably not worth the loss of speed and the extra programming effort.

Running Time. The running time for this algorithm is relatively fast. On a 2.8 Ghz Pentium 4, the test program will find 1,000 loops in a dense grid in 175 milliseconds and 2,500 loops in about 800 milliseconds. Considering that Lechuguilla Cave has about 1,800 loops, these are very acceptable times. In mathematical terms, the time is low order polynomial and is as good or better than other algorithms in the literature [4, 5, 6]. (Time in milliseconds = $m \times n \times k_1 + k_2$, where $k_1 = 1.29E\text{-}4$ and $k_2 = 0.08$.)

Loop Closures. One point should be clarified. There is a distinction between finding loops and closing loops. You might assume that in order to close loops, you would have to find loops. Actually, that is not the case. Closing loops only requires that you find junctions and traverses, which is a fairly easy task. The primary value of finding minimum cycles is to help locate blunders.

Test Software. In order to experiment with different loop finding techniques, I built a test program that is capable of generating and displaying large sets of random and non-random loops. It will find and display minimal loops using this and other algorithms. It will also display various kinds of spanning trees and you can even save graphs for later analysis. For those who are interested in experimenting with graphs, spanning trees and minimal cycles, the program can be down loaded here:

http://www.fountainware.com/download/loopfinder.zip

There is no installation program, so you will need to copy the files to a folder and create whatever shortcuts you need. There is a brief help file that explains most of the controls and options. Sources to the main algorithms are included.

Bibliography:

1. Algorithms for generating fundamental cycles in a graph.
Authors: N. Deo, G. M. Prabhu, and M. S. Krishnamoorthy
ACM Trasn. Math. Software, 8:26-42, 1982.

2. A faster algorithm for Minimum Cycle Basis of graphs.
Authors: Telikepalli Kavitha, Kurt Mehlhorn, Dimitrios Michail, and Katarzyna Paluch
The 31st International Colloquium on Automata, Languages and Programming
July 12-16, 2004, Turku, Finland

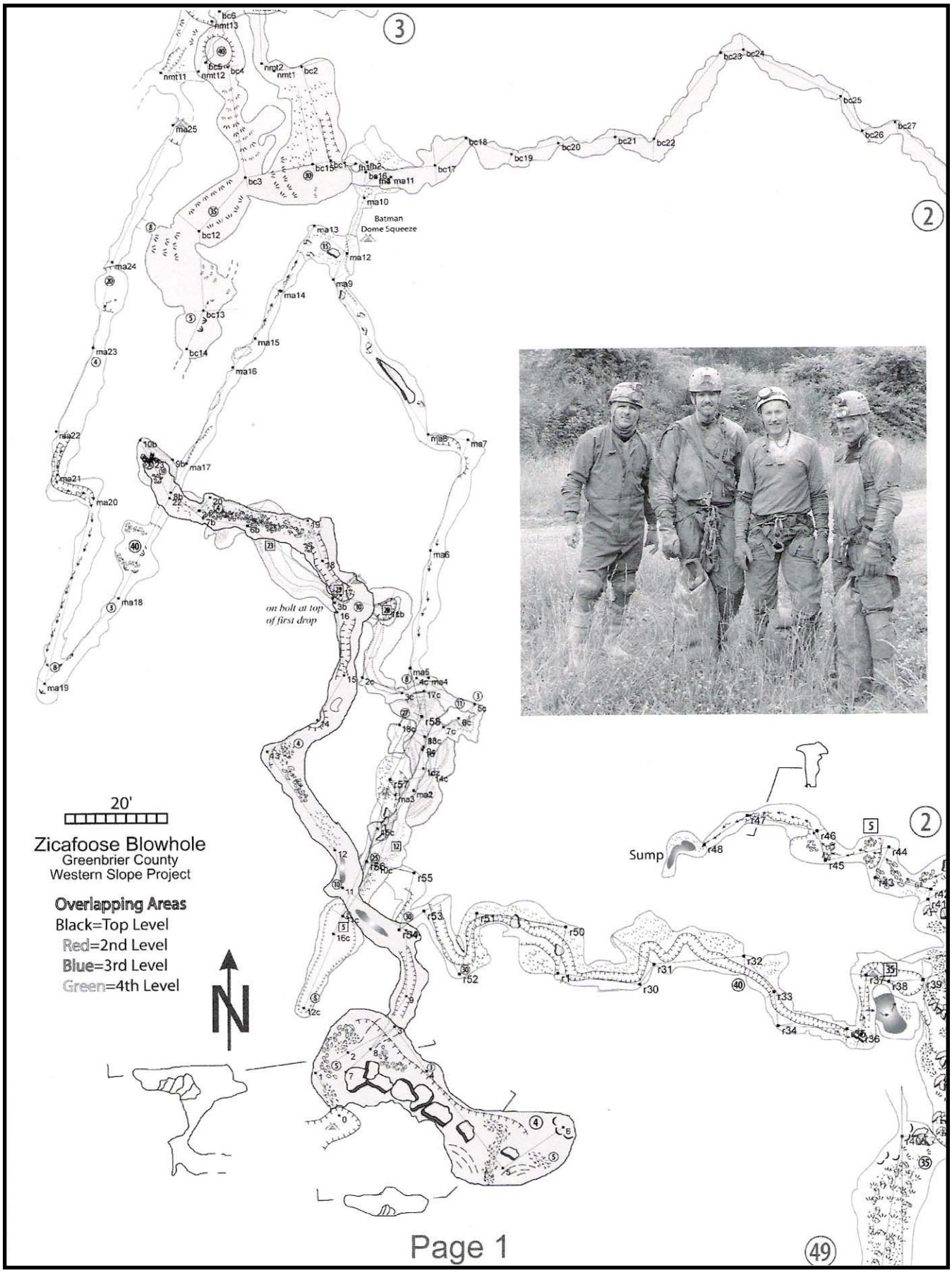
3. Implementation of algorithms for sparse cycle bases of graphs.
Author: Manuel Huber
Electronic Edition Available Here:
http://www-m9.ma.tum.de/dm/cycles/mhuber/

4. Implementing Minimum Cycle Basis algorithms.
Authors: Kurt Mehlhorn and Dimitrios Michail
WEA 2005: 4th International Workshop on Efficient and Experimental Algorithms
May 10-13, 2005, Santorini Island, Greece

5. Application of Shortest Path Methods.
Author: J.C. de Pina. PhD
thesis, University of Amsterdam, Netherlands, 1995

6. A polynomial-time algorithm to find a shortest cycle basis of graph.
Author: J. D. Horton.
SIAM Journal of Computer, 16:359-366, 1987.

7. Graph Theory
Author: Reinhard Diestel
Springer-Verlag New York 1997, 2000
Electronic Edition Available Here:
http://www.math.uni-hamburg.de/home/diestel/books/graph.theory/index.html



from the Zicafoose Blowhole Cave Project PDF Map book by Mark Passerby, 2004