Interoperability between the StraboSpot graph database and GIS software– A Malpais Mesa Use Case

By

Emily Bunse

B.Sc., Northwest Missouri State University, 2015

Submitted to the graduate degree program in Geology and the Graduate Faculty of the University of Kansas in partial fulfillment of the requirements for the degree of Master of Arts.

Chair: Dr. J. Douglas Walker

Dr. Leigh Stearns

Dr. Michael Taylor

Date Defended: 21 August 2018

The thesis committee for Emily Bunse certifies that this is the approved version of the following thesis:

Interoperability between the StraboSpot graph database and GIS software– A Malpais Mesa Use Case

_____

Chair: Dr J. Douglas Walker

Date Approved: 29 August 2018

Abstract

Geographic Information Systems (GIS) have been used by field geoscientists for decades to digitally collect data with several benefits: observations can be made in the field at a map independent scale while using multiple basemaps, the burdensome task of digitizing handwritten field notes and maps back in the office is eliminated, and the use of global positioning systems (GPS) has led to more precisely located observations. Despite the numerous advantages of using GIS in the field geosciences, the lack of agreement upon a standard schema for relational database management systems used in GIS along with new geospatial technologies and the popularization of mobile applications has led to the development of a novel geologic data management system, StraboSpot.

StraboSpot consists of a mobile application available for Android and iOS devices for data collection and an online graph database for data storage, management, and sharing. The issue of schema creation has been solved with the development of a lexicon through contributions from the structural geology and tectonics, sedimentology, petrology, and microstructural communities and the utilization of a graph database for storage, which is schema-less by definition. Users of StraboSpot can collect, store, and share their geologic data, making it an all-in-one solution for geoscientists to publish their data using open source techniques. Data is stored in a users' StraboSpot account which can hold multiple StraboSpot projects consisting of multiple datasets containing Spots. A Spot is a point, line, or polygon containing a set of observations over a user-defined spatial extent. The spatial extent of a Spot can be in real world coordinates (when set using maps or georeferenced aerial imagery), pixel coordinates (when set using a photo) or other systems as needed. Tags, "sticky-note"-like categories, can be used to conceptually group Spots.

StraboSpot does not have the cartographic and analysis tools found in a GIS, so it became necessary to establish connections between StraboSpot and commonly-used GIS software, ArcGIS and QGIS. GIS connections – an ArcMap Add-In and QGIS Plug-In – have been designed and programmed which have download and upload capabilities with StraboSpot. Download or upload of a StraboSpot project and dataset occurs through deserialization/serialization and parsing of JSON (Java Script Object Notation) and GeoJSON transferred between the GIS and StraboSpot's graph database via RESTful communications.

I traveled to Malpais Mesa, Inyo County, CA in October 2016, accompanied by two other University of Kansas students working on StraboSpot, to beta test StraboSpot's mobile app. Malpais Mesa was chosen due to the complexity of the area – it is situated in the Eastern California Shear Zone and the westernmost range of the Basin and Range province – which would adequately test for bugs in the mobile application and create robust StraboSpot data which I then used in the development of the GIS connections. Most importantly, it was an excellent location to compare the geoscientist's workflow with the capabilities and structure/interface of StraboSpot.

The interoperability between StraboSpot and the GIS connections provides users with a user-friendly and seamless method of downloading data collected in the field, performing various analyses such as running topology, and then uploading the "cleaned up" data for further use in StraboSpot. Geoscientists will also be able download all their StraboSpot data and create professional map products using the cartographic layout tools in GIS. The raw data behind those map products will be easily shared through StraboSpot leading to greater transparency, reproducibility, and reuse of geologic data.

Acknowledgements

I would like to thank the entire StraboSpot team, in particular:

Jason Ash for patiently teaching me and writing code snippets for handling RESTful web services and GeoJSON, all the assorted programming advice, and reviewing this manuscript.

Carson Rufledt and Katie Graham for accompanying me out to Malpais Mesa- thanks for bringing your adventurous attitudes and for helping wrangle all those iPads!

Doug Walker for supporting me through this project with resources, funding, input, advice, and encouragement.

Clay Campbell for creating a wonderful dataset at KU Field Camp 2016 which led to many bug/logic fixes in the Add-In and figures.

Additionally, Kristen Jordan-Koenig at the Kansas Geological Survey for getting me started on the Add-In and giving lots of advice, input, and encouragement along the way as well as reviewing this manuscript.

And finally, my loving family and friends who have supported and encouraged me. In particular, my father whose reverence for the Earth and love of maps helped shape me into the person I am today and my fiancé, Tyler, for always being my rock.

StraboSpot is funded under National Science Foundation grants to Dr. Julie Newman, Dr. Basil Tikoff (EAR 1347285, 1639549, and 1639748), and Dr. J. Douglas Walker (EAR 1252279 and 1347331, and ICER 1639734 and 1639738). Funding from Dr. J Douglas Walker's grants funded the Malpais Mesa field work.

Table of Contents

Page

Chapter I: Introduction

The Paper Map Problem

In Roger Tomlinson's 1974 groundbreaking and discipline-defining PhD thesis on the creation of the Canada Geographic Information System (GIS), he begins by describing two major disadvantages of using manual cartographic techniques:

1.) The challenge of fitting all useful data within the confines of one physical graphic while mitigating detail loss or generalization of the data, and

2.) The time and knowledge constraints the use of physical maps puts upon the user, i.e. the reading and analyzation of maps requires time and an understanding of the map format and purpose (Tomlinson, 1974).

Tomlinson played an integral role in creating the world's first GIS software, which addressed the above two issues with paper maps by using computers to map and analyze data for many different purposes relating to land management. One such purpose GIS software has become utilized for is the collection and curation of geologic data. However, as Tomlinson (1974) also mentions in his thesis, "The tools for handling data also have not been developed for their own sake, but rather have grown pragmatically out of institutional need." This is heavily evidenced in the geosciences with the introduction of ruggedized laptops and various handheld computers/PDAs to collect data in the field beginning in the late 1990s and early 2000s when portable devices which were GIS-enabled were introduced to the market (Walker and Black, 2000; Pavlis, 2010). These devices became even more powerful when the global positioning system (GPS) became available for civilian use in 2000 (Whitmeyer et al., 2009). The addition of these devices to the field geologist's tool belt was a major leap forward from collecting data though paper mapping, which had been the undisputed method of collecting and publishing

geologic data since William Smith's publication of a geologic map with accompanying cross sections of England and Wales (Whitmeyer, 2009). The addition of GIS to geologic field work was revolutionary in allowing scale-independent collection and analyzation of data (Walker et al., 1996) as well as freeing the geologist from the tedious job of digitizing observations made on paper once they were out of the field (Walker and Black, 2000, McCaffrey et al., 2005). Since the creation of geologic maps relies heavily upon the geologist's interpretation of collected data, digital collection of data allowed geologists to think beyond the data specific to the final product, a paper map at a specified scale, since the raw information could more easily be provided to the user of the paper map (Jones et al., 2004). This meant data collected became less generalized (bound to a specific map scale) and the viewing of the data more interactive (Loudon, 2000). Despite these advantages, there has been little consensus on standardization and schema design of GIS methods in the geosciences.

Standardization

The lack of standardization in geologic data management systems led to challenges in sharing data with peers due to differences in data formatting preferences (e.g. shapefiles, Excel tables, and XML documents) and proprietary license issues. For example, ArcGIS is popular in the geosciences, but some cannot pay the licensing fees to access the full software suite. Geoscientists soon saw the need to develop a one-size-fits-all (or at least 'most') data management system to encourage collaboration and sharing of geoscience data and sought to develop a mobile system to collect, store, and share geologic data. During an NSF EarthCube field excursion where participants explored the merits of various data collection tools and methods (which I attended- see Fig. 1), it was concluded that a community-developed app for mobile devices which could be used on a variety of mobile devices (i.e. smart phones and

tablets) was the best course of action to spark conversations within the geoscience community about the development of standards for data and metadata.

Fig. 1: In Fall 2015, I attended NSF EarthCube's Earth-Centered Communication for Cyberinfrastructure (EC3) field excursion which brought together geologists and cyber professionals. We took field excursions to test many devices. Pictured are: ArcGIS software on a Panasonic Toughbook; a clipboard with paper mapping materials; ArcGIS enabled GPS PDA; and a tablet with apps like: StraboSpot, FieldMove™ Clino, Strike and Dip™, GeoCompass™, and many more. Photo is the author's own.

There were concerns regarding the shortcomings of current hardware such as the challenge of viewing a screen, even on the brightest setting, on a bright day and problems with using touchscreen devices in the rain. The group also concluded that such a data management system needed to be open-source which rules out systems which rely heavily on ArcGIS and other proprietary software (Mookerjee et al., 2015).

The geosciences will benefit greatly from an open-source platform which allows for the collection, management, and sharing of field collected data. When data is easily available, it has a greater chance of being reused for other purposes which will add value to data collected leading to a reduction of costs for future projects. Greater reproducibility is another advantage of open-source acquired geoscientific data, however, some data in the geosciences such as

earthquakes and eruptions are unique and should be preserved. Finally, readily available data can also lead to greater public transparency and aid decision-makers (Fowler, 2016).

Schema

Use of a GIS presents challenges due to the set up required to create a structure for the data (schema) upfront since GIS data is stored in a relational database management system (RDBMS). The concept of KISS (keep it simple, stupid) is commonly used when evaluating geologic data systems and has been employed in the creation of various schemas for geologic data, which have made the use of a GIS for geologic data entry and management more efficient than handwritten notes and maps (Pavlis, et al., 2010). This advantage over handwritten data collection was largely due to the automatic capture of location, keeping the focus on the geospatial context of the data being collected. Additionally, GIS let users collect data using different basemaps and aerial images at varying map scales (e.g. 1:24,000 versus 1:63,360) (McCaffrey et al., 2005). Examples of schema for geologic data in a GIS can be found throughout the literature, most notably the United States Geological Survey National Cooperative Geologic Mapping Program (USGS NCGMP) Geologic Map Schema (GeMS) which is the standard for formatting data used to publish geologic maps (USGS NCGMP, 2018). In order to help users make their data NCGMP compliant, programmers at the USGS developed a set of ArcGIS Python Script Tools for creating and manipulating geologic data according to the GeMS schema. Examples of the geodatabase structure recommended for GeMS are found in Figures 2 and 3 and demonstrate the level of complexity that a robust relational geologic database management system requires.

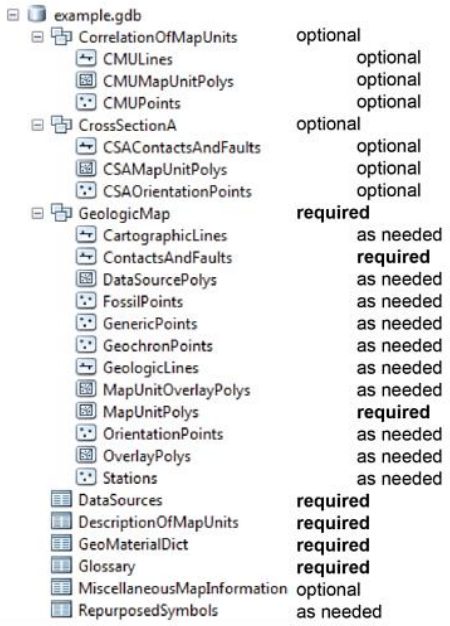


Fig. 2: An example GeMS geodatabase containing points, lines, polygons, and text descriptions. Annotations on the right note the required data needed to be NCGMP compliant (a distinction used when State Geological Surveys submit maps under the STATEMAP program) (taken from USGS NCGMP, 2018).

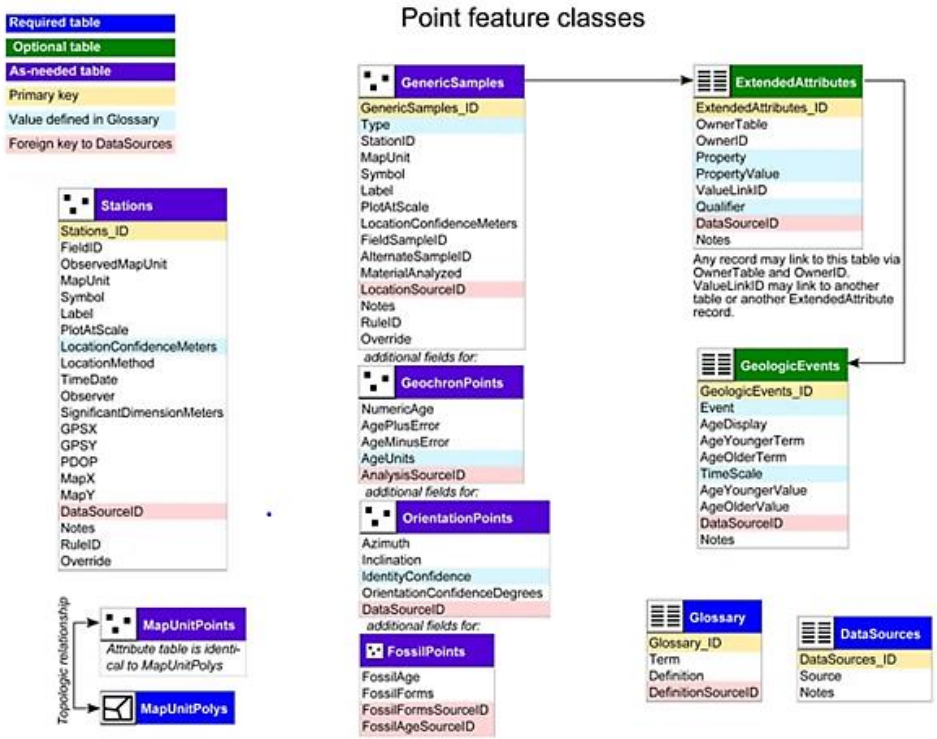


Fig. 3: Example of GeMS tables for managing geologic point features (taken from USGS NCGMP, 2018). Note that using this schema for data collection in the field would require navigating multiple feature classes and some projects will not need certain tables. However, this database schema focuses on encouraging the curation of robust metadata which will be crucial for interoperability in a national geologic map database.

While open-source GIS systems, such as QGIS, and mobile GIS apps are available (examples include: ESRI's ArcCollector™, Mappt™, and Cadenza™) the use of a GIS for geologic data collection is no longer the only method of geologic data entry and management due to new advances in geospatial data organization which will be discussed throughout this thesis. As there is a greater call for open-sharing of data in the geosciences, it is difficult to design a schema which fits an individual geoscientist's purpose yet can also be universally shared and integrated into other database systems. An additional challenge in schema design occurs when a geoscientist's project demands the use of data at radically different scales. An example of this is performing thin-section level analysis on rock samples taken during a field excursion. A final consideration is how to incorporate multiple observations at one location which may be of the same type. An example of this problem, which will be later referenced, is organizing several orientation measurements taken along a single fold (Walker et al., submitted).

While adoption of GIS in the geoscience projects from start to finish has served to mitigate the "paper map problem," GIS is no longer the only data entry option for geoscientists. Through development and adoption of an open-source geologic data management system which does not come with the challenges inherent with a GIS, the geosciences will address many issues that have been encountered throughout the process of evolving field collection techniques from paper mapping to GIS-based ones. Advantages of such a system are highlighted below through discussion of the development of the StraboSpot system.

An Overview of StraboSpot

The StraboSpot system, an NSF-funded, multi-university project, has been developed for the collection, storage, and sharing of geologic data in response to the need for a community-created, open-source geologic data management system. The StraboSpot mobile app is available

for both iOS and Android devices and allows for collection of geologic data, regardless of Internet connectivity, making it suitable for any field setting. StraboSpot projects and datasets can be uploaded to and downloaded from the StraboSpot database, a Neo-4j graph database, via the app. Once a dataset has been uploaded to a StraboSpot user's account, the user may choose to make their data "Public" which means the data is added to the StraboSpot Search Interface allowing other users to view and query all publicly available data and see who owns the data they find. Thus, StraboSpot fulfills the need in the geosciences for an easy, open-source solution for collecting and publishing geologic data.

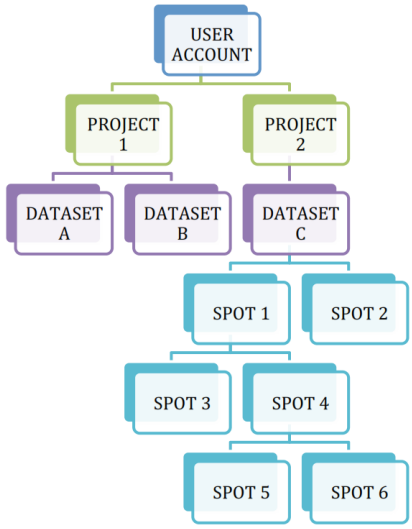


Fig. 4: The data hierarchy of a user's account in the StraboSpot system (taken from Rufledt, Walker, et al., 2017). Note that Spots can be 'nested' within other spots and a project (for example 'Field Camp 2018') can contain multiple datasets.

Data collection in the StraboSpot system, performed through use of the mobile app, is based off the concept of a "Spot" which works similarly to mapping by station where the user inputs a set of observations that are defined by a spatial extent. The flexibility of the Spot concept allows for collection of a variety of data at all scales. Spots are stored in a dataset and many datasets can be stored in a project on a user's account (Fig. 4). Unlike other digital data systems based on storage in a relational database, StraboSpot's graph database allows for rich and complex data entry so users can easily capture relationships among data (Fig. 5). Complex relationships are difficult to document in a GIS except through multiple table joins. StraboSpot also incorporates images of all types (photos, sketches, etc.) that can be used as basemaps.

There are two ways of organizing Spots: spatially, through nesting where Spots covering a smaller area are "nested" inside Spots spanning a larger area and conceptually, through the assignment of Tags which are quick, "sticky-note"-like categorizations of data- one example is "geologic unit". Due to this flexibility, a Spot can be a point, line, or polygon having either real world or pixel (image basemap) coordinates and a StraboSpot dataset can hold any combination of these types of data.

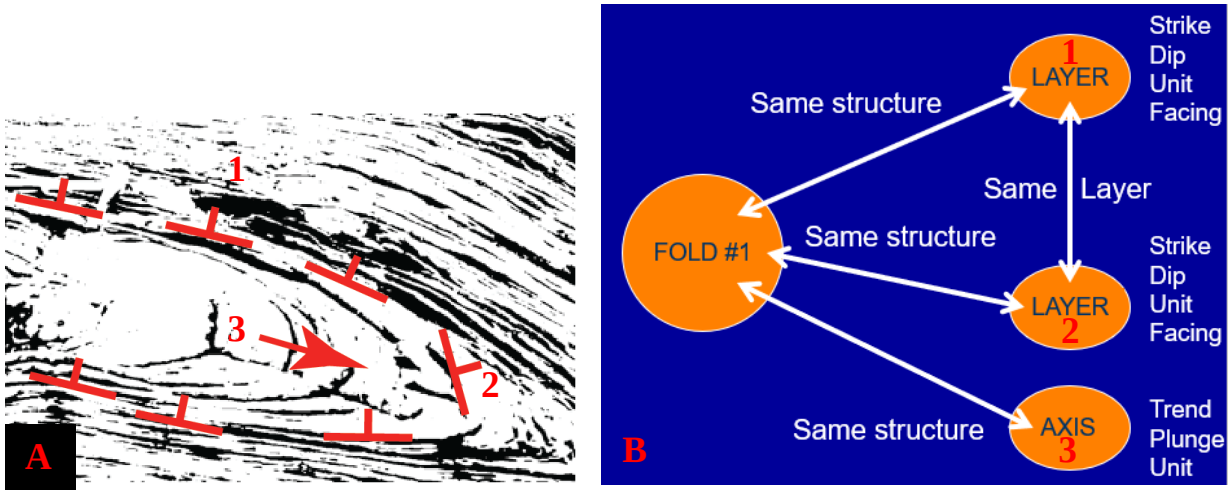


Fig. 5: Upon encountering the fold pictured in (A), a geologist using StraboSpot can establish relationships, as depicted in (B), between various orientation measurements which were recorded in the same rock layer and between the overall structure (Fold #1) and each observation/measurement. The "Fold #1" Spot is a Nest which contains the Spots for each orientation measurement and layer information. This could be accomplished in 'map view' or by taking a picture of the fold and dropping Spots directly on the image (taken from Newman et al., 2015).

The StraboSpot system has been developed by a team of geologists (PI's on the project include: Dr. J. Douglas Walker at University of Kansas, Dr. Basil Tikoff at University of Wisconsin-Madison, and Dr. Julie Newman at Texas A&M) in conjunction with programmers familiar with the needs of the geoscience community. The lexicon and workflow of the app and database has been developed through thoughtful discussions with members of the Structural

Geology and Tectonics, Sedimentology, Petrology, and Microstructural communities. Throughout my thesis work, I have contributed to the development of StraboSpot as part of the team. My contributions to StraboSpot are detailed in the next section.

Contribution of Thesis

The StraboSpot app is a powerful tool for data collection, but since the data analysis and cartographic construction tools that many geologists use were already offered by GIS software, the StraboSpot team sought interoperability between the StraboSpot system and two popular GIS software programs: ArcGIS and QGIS. Such tools or methods include: running topology, interpolation and calculation analysis, cartographic tools for creating map layouts, etc. As evidenced above, geologists have been using GIS software for data collection and analysis for decades, further necessitating a connection for the sake of familiarity. Therefore, this thesis provides background and details the methodology developed for extracting StraboSpot data and putting it into a GIS, editing the data, and uploading changes made in the GIS back to StraboSpot.

Considerations when choosing the approach for both ArcGIS and QGIS include: ease of use (i.e. the construction of concise, but clear workflows) and organization of the data in the GIS on both the geodatabase and table levels that facilitate a straightforward upload process yet is usable for the purposes of the user within the GIS. Due to the complex interactions between ArcGIS and StraboSpot, a VB.NET Add-In (as opposed to a Python Script Tool or Python Add-In) with download and upload capabilities was developed. For the sake of continuity, a QGIS Python Plug-In with a similar graphic user interface (GUI) to the Add-In was also developed. These GIS connections translate the StraboSpot graph structure to a relational structure and

provide a user-friendly, quick set-up of StraboSpot datasets in ArcMap and QGIS as well as a method for uploading edits made in either GIS software back to StraboSpot for further use. A concise explanation of the code will be found in the Methods section.

Throughout the thesis, several major challenges with development and programming of these GIS connections will be addressed. The first challenge dealt with flattening the JSON/GeoJSON (JavaScript Object Notation) objects which compose StraboSpot Projects and Datasets to fit the relational database model, yet remain useable and readable to the average geoscience GIS user. Flattening was a challenge which persisted throughout all levels of data management from geodatabase structure to the construction of individual table rows. Converting Spots to table rows was especially challenging since StraboSpot's database works based on graph theory. Graph databases are schema-less and operate through an additive method of storing data, meaning new types of data can be added at any time without disturbing the current operation of the database (Robinson, Webber, Eifrem, 2015). Relational databases require a schema to be built upfront with only minor changes allowed after creation. In the Literature Review, the sparse publications detailing the conversion of JSON to rows and columns in a table will be explored. Additionally, rewriting JSON files for use in ArcGIS and QGIS required different approaches.

Determining the role images from the StraboSpot dataset would have in the GIS was also a process. StraboSpot, as an all-in-one solution for collection of geoscience data, eliminates many of the tools geoscientists need in the field, one of which is a camera. Since the mobile app links directly to the camera in the user's device, many images can be added easily to Spots. The user can then map on those images as image basemaps, dropping point, line, or polygon Spots with the same types of data as map-scale Spots. The only difference between the Spot types is

image basemaps have pixel coordinates whereas map-scale Spots have real-world coordinates associated with them. GIS software can display such images as raster datasets in an unknown coordinate system, but to map data – for example, a point with orientation data or a line defining a contact between beds – on those images is a challenge.

In order to produce a robust StraboSpot dataset for development and testing of the GIS connections and participate in Beta testing of the app, I was accompanied by two other University of Kansas geology students with a dozen Apple iPad Mini 4 tablets to investigate Malpais Mesa for one week in October 2016. The study area is in the westernmost Basin and Range Province, just east of Owens Valley (westernmost basin) and in the southernmost portion of the Inyo Mountains (westernmost range). The Malpais Mesa datasets will serve as the examples for how StraboSpot data can be effectively translated and organized for use in a GIS. A geologic background will be given in the Literature Review section and screenshots of the data from ArcGIS will be shown in the Results section.

Chapter II: Literature Review

To fully understand the design and code structure of the StraboSpot/GIS Connections which will be detailed in the pseudocode in the Methods section, the following code components should be understood: graph database structure, RESTful communication through HTTP, GeoJSON, and the use of ArcGIS Toolboxes.

StraboSpot's Graph Database

There are currently a wide variety of databases available to store data, but the many possibilities can be divided into two subgroups based on how users interact with them: those which use Structured Query Language (SQL) and those which do not – NoSQL. SQL is used to

query relational databases which are tables made up of rows and columns. Relational databases work based on a set-theory and are named since they use relational algebra to recover information. Relations, the tables in the database for which the database type is named, are made up of tuples, i.e. lists, that hold attributes and their values; these are the rows in the table. Columns are named by a tuple of fields which include a name and a certain data type such as string, integer, Boolean, etc. for each field (Redmond and Wilson, 2012). Relational databases, which tend to be the traditional database choice, are quite powerful when used in applications where the schema is known when the database is created. One type of NoSQL database which is designed to organize data via the relationships among data without needing any schema defined upfront is a graph database.

The lack of any schema in a graph database, allows it to accept into its "structure" any type of data the user adds. Where relational databases are symbolized through tables, graph databases are considered "whiteboard friendly" (Redmond and Wilson, 2012), meaning that one could draw a graph model as a brainstorming web graphic. Graph databases work by building relationships called "edges" between groups of data called "nodes". Querying is done through traversal of these nodes and edges. While graph theory has existed for hundreds of years, graph databases were not developed until the 1990s. Graph databases have become popular for housing large quantities of data which need to be rapidly queried, making them ideal for powering most social media and online commerce sites (Robinson, Webber, Eifrem, 2015). Therefore, a graph database is an excellent choice for geologic field data storage because it can hold any combination of spatial data types (e.g. point, line, polygon) within one dataset and show the relationships between data as demonstrated in the Fig. 5 fold example. Another advantage to

using a graph database for storage of StraboSpot data is the rapid querying ability of complex data offered by such a database. See Fig. 6, after Robinson, Webber, and Eifrem (2015) for comparisons between a relational database and graph database when relationships between data are crucial to query. Vicknair, et al. (2010) found similar querying results when the Neo4j graph database and MySQL relational database were compared based on querying speed and other subjective factors like support, maturity, ease of programming, flexibility, and security.

Depth (# of connections separating data)	RDBMS Query Execution Time (sec)	Neo4j Query Execution Time (sec)	Records returned by query
2	0.016	0.01	~2,500
3	30.267	0.168	~110,000
4	1543.505	1.359	~600,000
5	Unfinished	2.132	~800,000

Fig. 6: Robinson, Webber, Eifrem's 2015 manual for their Neo4J graph database, the brand of graph database utilized by the StraboSpot system, features numerous comparisons between relational (RDBMS) and graph databases to familiarize new users with graph database theory. Execution time of queries with complex depth of relationships is a major difference in the two databases. The table above, after Robinson, Webber, Eifrem, (2015), displays results for returning queries to find extended friends. In a relational database this relationship is found through querying multiple tables which are joined and tends to fail once the query is extended a depth of five table joins (or friend-of-friend-of-friend-of-friend-of-friend) but the graph database sees hardly any performance change to perform the same type of query.

StraboSpot's Mobile App

While StraboSpot's graph database does not have a defined structure for data, the team is able to define and add to a lexicon of geologic terms. These lexicons are built by conferencing with professionals in each geologic subdiscipline which StraboSpot is working to incorporate into the system. Recommendations from the Structural Geology and Tectonics subdiscipline

have been fully incorporated and work is underway in the Sedimentology, Petrology, and Microstructures subdisciplines. From these lexicons, pages for the app are developed by the geologists in an online utility called KoBoToolbox, which uses xls-forms, and then ported into the code (see Fig. 7 for examples of forms). The visibility for these pages can be turned on/off as the user needs within the app.

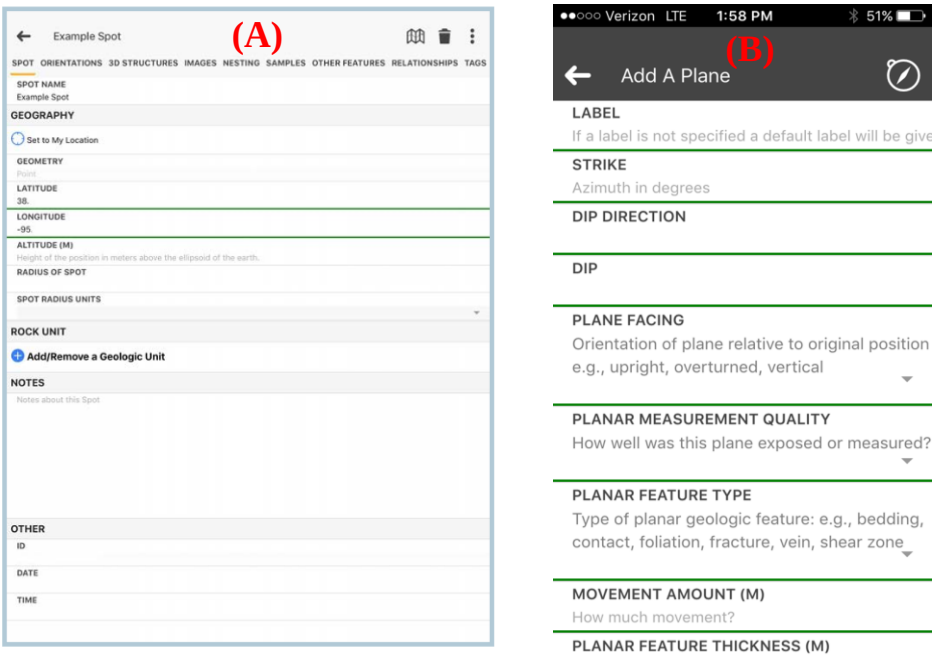


Fig. 7: Examples of forms StraboSpot users can fill out in the mobile app. (A) is the basic form that opens when users place a Spot on the 'map view' or they click '+' in the Spot List. From the main Spot Page, users can navigate to other pages like 'Orientations' where they can add a planar measurement (B). The app features a built-in compass for taking such measurements or users can add info manually. StraboSpot features Tagging, an easy method of conceptually categorizing Spots. The most common type of Tag is a Geologic Unit, shown in (C). Users fill out information about a unit (age, rock type, etc.) then can easily assign the Tag to other Spots by choosing it from a list under the Spot's Tags page. Images taken from Rufledt, Walker, et al. (2017).

The schema-less structure of the graph database and pre-determined fields in the app will save the user time spent creating the database they will use for data input in the field. The user simply inputs information into the StraboSpot app to set up their project and dataset(s) and adds and/or downloads any map tiles they might require in the field (see StraboSpot help

documentation, https://strabospot.org/help, for more detailed set up directions). The only caveat of using the StraboSpot system for download and upload of data in a GIS is that any non-StraboSpot fields added to the GIS and uploaded to the GIS tables will be stored in the graph database, but will not be accessible, currently, within the app.

ARC/INFO, ESRI ArcView, and ArcGIS have been used for collecting geologic data in the field since the early 1990s (Pavlis et al. 2010), but when compared to StraboSpot limits the user in the type of data which can be added and requires lots of time upfront for database design. Also, since an ArcGIS database is a relational database, recording relationships between data across tables is challenging and table joins are the only meaningful option. Before a field excursion, the user must set up the entire data structure of the geodatabase, which is difficult to change later due to the rigid nature of relational databases. For this reason, geologic reconnaissance excursions either require complex geodatabase systems with every possible geologic structure, measurement, and observation type included in the list of fields across several feature classes to accommodate any data type encountered (point, line, or polygon), or the geoscientist must take the more generalized option of using a basic geodatabase with a note-taking section to record data. Adding drawings and photos to this database is also complicated, requiring careful documentation of the photo in the field and time back at camp to add the image files to the database. There are GIS apps available, such as ESRI's ArcCollector™ which can connect to images on the device, but these apps still pose the problem of relational database schema creation upfront. For more expert users or those already familiar with their field area, this system can work quite well and can produce geologic maps with more detail and accuracy than traditional paper mapping with additional data recorded in a notebook.

It is my experience from the field excursion to Malpais Mesa that having access to StraboSpot greatly improved the speed of mapping in the field. For example, in comparison to paper mapping, it was much simpler to check the location the GPS plotted our position than to plot the position on a pre-drawn grid. We were able to locate ourselves more precisely and quickly using the StraboSpot app. Additionally, the rock types found at Malpais Mesa were quite diverse and using image basemaps and Tags to organize what we were viewing was an intuitive method. It would have been challenging to input structural and stratigraphic relationships among layers in a relational database environment as quickly.

REST and HTTP

Interacting with the StraboSpot database server requires coding RESTful (Representational State Transfer) communications with StraboSpot implemented through HTTP (Hypertext Transfer Protocol). REST architecture was developed in the early 1990s in response to the need for a better method of network communication which was intuitive at all levels of development (Fielding, 2000). The creation of REST, HTTP, and URIs (Uniform Resource Identifier) contributed to the advent of the modern Web. To define REST services in the context of StraboSpot: "REST components communicate by transferring a representation of a resource…," (a representation of a user's StraboSpot Project, Dataset, or an individual Spot) "…in a format matching one of an evolving set of standard data types…," (JSON or GeoJSON are the data types sent), "…selected dynamically based on the capabilities or desires of the recipient and the nature of the resource." (Fielding, 2000). This transfer of information is between a client machine, the end making a request – the GIS connection on a user's computer – and a server machine, the end that waits for requests to be made and responds to the client –

StraboSpot. The transfer is ‘stateless’ in nature since all the information needed for the server to understand the request is within the request (Fielding, 2000).

StraboSpot data is a ‘resource,’ anything mappable using a URI, on the StraboSpot server which the Add-In is seeking representation of during download and is modifying the representation of on the server during upload. The common method of interacting with a set of URIs is through HTTP which does not necessarily require the user to have knowledge of an API (Application Programming Interface), (Wilde, 2007) however StraboSpot does have an API of controllers.

HTTP has four different methods for data to operate when performing a REST call: GET, POST, PUT, and DELETE (Feng et al., 2009). These methods correspond with the methods used to work with a graph model: Create (PUT), Read (GET), Update (POST), and Delete (Robinson, Webber, Eifrem, 2015). The GET and POST methods are used by the GIS connections to interact with StraboSpot data. When included as the method in a REST request, GET obtains the current state of StraboSpot data and provides the representation of that data through a data stream (so, GET does not change anything about the state of the resource on the server). GET is an important part of the GIS connections’ download code. The POST method “transfers a new state onto a resource,” (Feng et al., 2009), so REST calls including POST are important to the upload code since the user has altered the data and wants to impose that “state” onto the resource in the StraboSpot server.

The success of REST requests is communicated through status codes with numerical meanings. Those who have used broken website links have probably encountered one such message: 404: “Not Found.” However, in making requests of StraboSpot, the most common responses a GIS connection receives are 200: “OK” or “Created” and 400: “Bad Request.” Status

codes numbered 200-299 indicate successful execution of the REST request and 400-499 indicate an error in the request the client sent. Such a request should not be sent again without modification (Fielding, 2000).

The components of a REST request in the GIS connections include: the method of communication (GET or POST); the URI of the resource (any StraboSpot data on the server), the user's authentication information (email and password) in the form of a JSON stream, and, if performing a POST, the information needed to change the state of the resource in a JSON or GeoJSON stream. The anatomy of the URI is as follows: it begins with identification of the server's database, "https://strabospot.org/db," then is followed by the name of the controller describing the type of information in the database (examples include: "/datasetspotsarc" for the whole dataset, "/project" for the information about a project, or "/myDatasets" for a full list of the user's datasets), and the unique identifier of the resource, a Unix timestamp in milliseconds plus random digit assigned at the creation of the resource. A full list of the controllers can be found by visiting https://strabospot.org/api and will be referenced below in the Methods Pseudocode section.

GeoJSON

The data format for communicating about datasets on the StraboSpot server is GeoJSON; other communications are made with JSON. GeoJSON is a derivative of JSON (JavaScript Object Notation) that can store spatial information. JavaScript became a popular language for web interaction because of its low complexity, favorable security features, and execution style in relation to other languages on the Web like Java (Fielding, 2000). Web browsers natively support JSON formatted data meaning that GeoJSON is easily used in web applications (Bostock and Davies, 2013).

JSON is also popular since its data structure is human-readable (a set of key: value pairs; DiScala and Abadi, 2016) and more compact in file structure and easier to map to objects in most object-oriented languages (e.g. JavaScript) than XML (Chasseur et al., 2013). Data in JSON format undergoes encoding transformations called serialization and deserialization to translate JSON objects to human-readable text for processing and vice versa. The download code uses serialization to transform the JSON object sent by the StraboSpot server into a human-readable JSON-string document used to represent Spots or Tags. The upload code deserializes a string of JSON-formatted text containing the edited Spots into a JSON object that is sent in a data stream of bytes to the StraboSpot server.

GeoJSON has a specific structure to hold the data about geographic features that can be defined by objects, essentially containers of data which are surrounded by '{}', and arrays, lists of objects which are surrounded by '[]'. In the case of a StraboSpot dataset, the largest object is a 'FeatureCollection' type which contains an array of 'Feature' objects. One 'Feature'-type object represents a Spot and contains: one 'geometry' object and one 'properties' object. The 'geometry' object has two parts: the type of geometry ('point', 'linestring', or 'polygon') and an array of coordinate values. The 'properties' object is composed of the remaining data about the Spot such as 'name,' 'SpotID,' and 'date,' but can also contain named object arrays which correspond to the tabs in the StraboSpot GUI (Graphic User Interface- e.g. Orientation, Images, Inferences, etc.). Figure 8 shows a general sample of GeoJSON (Butler et al., 2016):

```
{
       "type": "FeatureCollection",
       "features": [{
           "type": "Feature",
           "geometry": {
               "type": "Point",
               "coordinates": [102.0, 0.5]
           },
           "properties": {
```

```
            "prop0": "value0"
        }
    }, {
        "type": "Feature",
        "geometry": {
            "type": "LineString",
            "coordinates": [
                [102.0, 0.0],
                [103.0, 1.0],
                [104.0, 0.0],
                [105.0, 1.0]
            ]
        },
        "properties": {
            "prop0": "value0",
            "prop1": 0.0
        }
    }, {
        "type": "Feature",
        "geometry": {
            "type": "Polygon",
            "coordinates": [
                [
                    [100.0, 0.0],
                    [101.0, 0.0],
                    [101.0, 1.0],
                    [100.0, 1.0],
                    [100.0, 0.0]
                ]
            ]
        },
        "properties": {
            "prop0": "value0",
            "prop1": {
                "this": "that"
            }
        }
    }]
}
```


Figure 8: An example of a GeoJSON FeatureCollection showing all three geometry types StraboSpot datasets can generate: Point, LineString, Polygon. Taken from Butler et al. (2016).

Note that data is stored as a series of key: value pairs. Keys correspond to fields on the forms displayed in the StraboSpot app and values are either automatically generated (in the case of ‘date,’ ‘id,’ ‘modified_timestamp,’ and various fields under ‘images’) or are values entered by the user (‘strike,’ ‘dip,’ ‘facing,’ etc.) from drop down menus or by typing text. Aside from

datasets, all other information passed between the Add-In and StraboSpot is in JSON format. This data includes user authentication and information about datasets and projects.

Once one understands the structure of GeoJSON, a file can be easily edited in any text editor (Bostock and Davies, 2013). Also, many GIS software packages, especially open-source ones such as QGIS, natively support the import and export of GeoJSON into geodatabases. Another advantage of GeoJSON is that it only accepts data in the World Geodetic System 1984 (AKA: WGS 84 or wkid: 4326) coordinate reference system (using latitude/longitude), making it compatible with data collecting techniques using GPS (like StraboSpot) to locate the user.

JSON, despite its popularity in web applications, has become a complex challenge to import into relational database management systems (RDBMS). In fact, in the literature, those who have been working on this problem recognize that document NoSQL databases such as MongoDB or CouchDB have been developed specifically to handle the denormalized (normalization is the reorganization of a RDBMS which helps to cut down on data redundancy) structure of JSON. MongoDB and Apache's Couch DB are both document databases but differ in how users interact with them. Mongo DB works best for curating large datasets residing in datacenters and has commands which are familiar to SQL. Couch DB is more flexible in deployment situations meaning it is lightweight, but powerful and works well with web technologies (Redmond and Wilson, 2012). In a document NoSQL database, JSON objects are stored as "documents" which are analogous to records in a RDBMS except the documents have no schema. "Collections" of JSON objects in a NoSQL database are analogous to RDBMS tables (Zhang et al., 2014).

NoSQL databases are not easily supported in GIS software, but as Zhang et al., 2014 shows, GIS data can be converted from shapefile to MongoDB using the pyshp and pymongo Python libraries. This approach can be good for handling "big data" but does not solve the issue of being able to handle JSON data in a GIS environment.

Another approach has been to bypass the use of NoSQL databases and figure out a way to bring order to the denormalized JSON data, so it may be stored in a RDBMS. RDBMS offer several advantages to users that NoSQL databases lack such as: declarative query language (NoSQL databases are defined as being "Not Only SQL," meaning they employ other methods of querying than SQL - structured query language), native joins of data, and ACID transactions (Atomic/all-or-nothing execution, Consistency, Isolation from other transactions, and Durability) (Chasseur et al., 2013). Approaches to organizing the data for table creation differ by how much analysis of the input JSON is performed. For example, the Argo method, one of the first attempts to translate JSON to a RDBMS, developed in Chasseur et al., 2013, decomposes the JSON objects by their key: value pairs and handles the nested structure by appending the name of the nest to the key's name. Their sample JSON is in Fig. 9:

```
{
	"name": "George Bluth",
	"age": 58,
	"indicted": True,
	"kids": ["Gob", "Lindsay", "Buster",
		{
			"name": "Michael",
			"age": 38,
			"kids": ["George-Michael"]
		}],
	"rival": "Stan Sitwell"
}
{
	"name": "Stan Sitwell",
	"age": "middle-aged",
	"charity_giving": 250120.5,
	"kids": ["Sally"]
}
```

Fig. 9: Sample pair of JSON objects used for illustration of RDBMS adaptation in the Argo method taken from Chasseur et al. (2013).

To preserve the differences in data types (strings, numbers, and Boolean values), separate columns (Argo/1, Fig. 10) and tables (Argo/3, where each data type – str, num, and bool – as shown in Fig. 10 are stored in separate tables which can be joined by the 'objid' field) were used for storing values. Values which belong to the same JSON object are grouped by an object id.

objid	keystr	valstr	valnum	valbool
1	name	George Bluth	NULL	NULL
1	age	NULL	58	NULL
1	indicted	NULL	NULL	true
1	kids[0]	Gob	NULL	NULL
1	kids[1]	Lindsay	NULL	NULL
1	kids[2]	Buster	NULL	NULL
1	kids[3].name	Michael	NULL	NULL
1	kids[3].age	NULL	38	NULL
1	kids[3].kids[0]	George-Michael	NULL	NULL
1	rival	Stan Sitwell	NULL	NULL
2	name	Stan Sitwell	NULL	NULL
2	age	middle-aged	NULL	NULL
2	charity-giving	NULL	250120.5	NULL
2	kids[0]	Sally	NULL	NULL

Fig. 10: RDBMS result of processing the JSON from Fig. 9 to rows and columns using Argo/1. Note that depth (nesting) is represented by extending the key name. After Chasseur et al. (2013).

The problem that arises when using the Argo method where data is only analyzed for data type is that the data are not as intuitive to query since keys are not treated as fields or column names. This makes table SQL queries more challenging.

Other attempts to normalize JSON data look at the input dataset with more detail. This is the solution I implemented to transfer data from StraboSpot to ArcGIS and QGIS. One example that is similar to the approach of the StraboSpot GIS connections was created by DiScala and Abadi (2016). Their work is in three phases where the first two identify and match entities and

the final phase analyzes the output of the first two phases to normalize the tables. For reference, a sample of their JSON is in Fig. 11:

```
{
      'submissionId': 1576,
      'title': 'Perils of DB',
      'author': {
            'email': 'js@e.pfl',
            'name': 'J. Snow',
            'institution': {
                  'name': 'EPFL',
                  'number': 1 }
            },
      'reviewer': {
            'email': 'dd@ca.uk',
            'name': 'D. Duck',
            'institution': {
                  'name': 'Cambridge',
                  'number': 14 }
            }
}
```

Fig. 11: Sample JSON data taken from DiScala and Abadi (2016) to accompany their tables shown in Figures 12 and 13.

The key to this method of organization is that nested attributes are handled by appending a prefix to the key denoting the name of the parent object to create one table (Fig. 12).

subId	title	authEmail	authName	authIName	authINum	revEmail	revName	revIName	revINum
1576	Perils of Db	js@e.pfl	J. Snow	EPFL	1	dd@ca.uk	D. Duck	Cambridge	14

Fig. 12: Table produced after the first phase of the DiScala and Abadi (2016) algorithm. The sample JSON object from Fig. 11 is the first row of this table. The most important part to notice is the prefix values in column names (i.e. 'authEmail' means the email key: value pair in the 'author' object). After DiScala and Abadi (2016).

Functional relationships between attributes are identified and are used in later phases of the algorithm to normalize the data into separate tables (Fig. 13).

subId	title	revID	authID	ID
1576	Perils of DB	3	1	1

email	name	instID	ID
js@e.pfl	J. Snow	1	1
dd@ca.uk	D. Duck	4	3

INum	IName	ID
1	EPFL	1
14	Cambridge	4

Fig. 13: The normalized tables of Fig. 12's table produced by the DiScala and Abadi (2016) algorithm. After DiScala and Abadi (2016).

The advantage of my being able to work directly with the StraboSpot API developer and manager, Jason Ash, and the knowledge of the StraboSpot lexicon led to an approach to parsing and organizing the JSON that was much like the DiScala and Abadi (2016) work. The method developed means that parsing of the JSON is very specialized for the StraboSpot system. For the sake of simplicity, I chose to put the impetus of normalizing the data downloaded on the user based on specific nests (e.g. creating separate feature classes for orientations, samples, images, etc.) since it is my opinion, upon extensive review of GIS schema used in the geosciences, that users have individual preferences for how their data is organized. Some users may want to normalize their data according to the GeMS schema, departmental or state survey schema.

ArcGIS Toolbox

The key to creating ArcMap layers during download from StraboSpot and exporting data from ArcMap during upload to StraboSpot is running various ArcGIS tools via code. ArcGIS tools can be viewed in ArcMap by opening the Catalog window and scrolling to and expanding the "System Toolboxes" tab in the file system. While there are a variety of tool types found there, the ones utilized by the Add-In fall under the "Conversion" and "Data Management" categories. The Conversion toolbox contains the two tools used to convert ESRI-style JSON into

a Feature Class, and the Data Management toolbox contains various tools for making tables, creating an image layer, and copying data to a feature class. To use an ArcGIS tool in code, I followed these steps from ESRI documentation for .NET developers: "1.) Add a reference to ESRI.ArcGIS.Geoprocessor. You may also need to add the ESRI.ArcGIS.Geoprocessing assembly if you want to use, for example, the result object or list datasets. 2.) Additionally, add a reference to the toolbox assembly to which the tool belongs. If you use more than one tool from different toolboxes, also add managed assemblies for those toolboxes. 3.) Create the geoprocessor object. 4.) Add the path to the custom toolbox if you are running a custom tool. 5.) Create a tool process object and set the parameter values. 6.) Call the Execute method on the geoprocessor." (ESRI, 2010).

Reliance upon geoprocessing tools within the code will enable greater ease when the Add-In is converted to run in ArcGIS Pro since the ArcGIS Pro API uses Data Manipulation Language (DML) only (https://github.com/esri/arcgis-pro-sdk/wiki/ProConcepts-Geodatabase). Geoprocessing tools will be the only method of implementing Data Definition Language (DDL) actions (database and feature class creation, etc.) in an ArcGIS Pro Add-In.

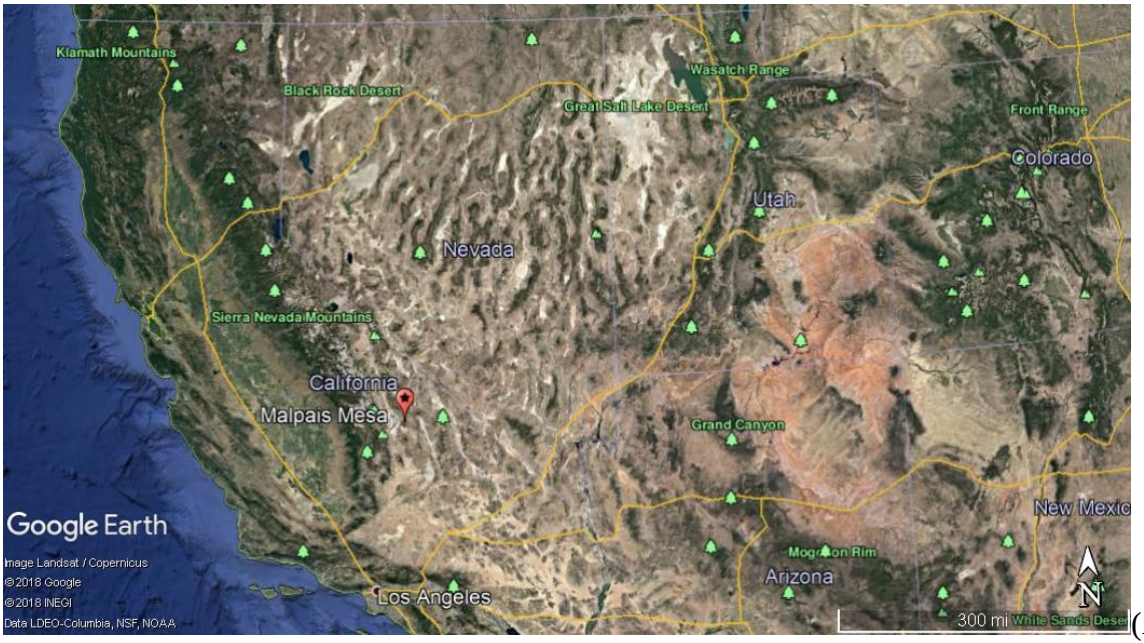
Note that while there are toolboxes in QGIS, since the software is open-source the API has Python functions which readily accept GeoJSON and turn the information into a vector layer. The StraboSpot GeoJSON still needs to be parsed to separate out Spots by geometry and to flatten the structure to achieve one key: value per cell in the table. Additionally, QGIS can accommodate many different geodatabases, so once a QGIS vector layer was created, I added options for the user to add the layer to a PostgreSQL or SpatiaLite database which was accomplished with Python functions.

Malpais Mesa Geologic Setting

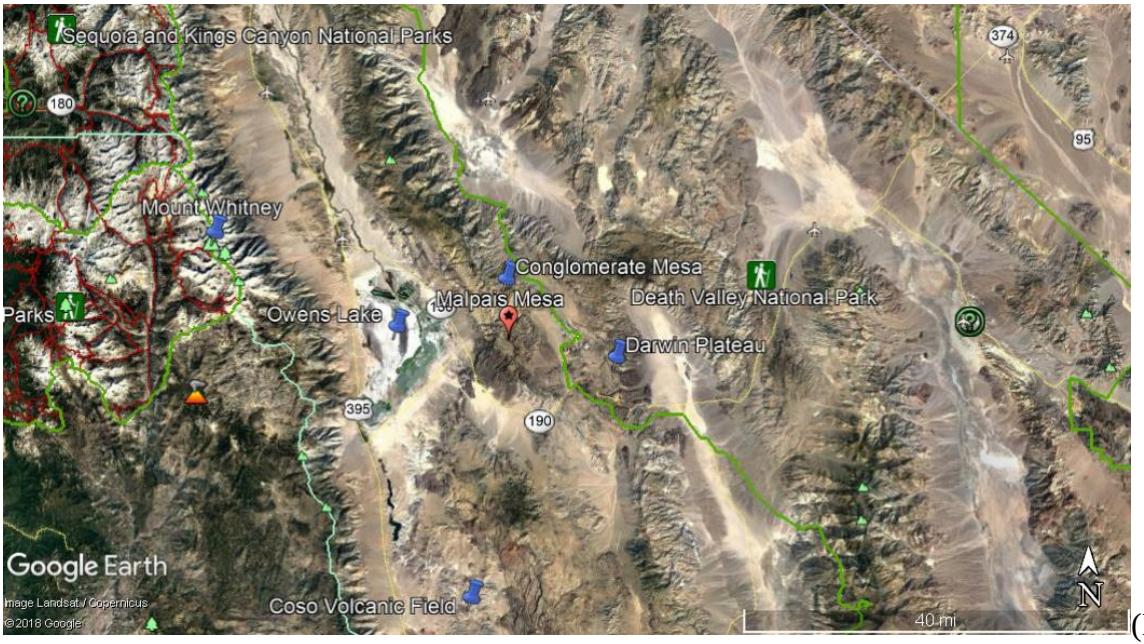
Malpais (aptly, Spanish for 'bad country') Mesa is located in the westernmost portion of the Basin and Range Province (Fig. 14). The field area was chosen for two reasons. First, I needed an area to collect data to evaluate the use of StraboSpot and test it as a field application. Second, Malpais Mesa is not well studied, and information gathered could lead to new conclusions about the significance of the area. Because of time limitation, however, the main focus of the work there was on using StraboSpot and not making new advances for regional geology. Even with the limited amount of new interpretations, I describe below the geologic setting in order to place the area in context.

Geographic Setting

Malpais Mesa is situated on the southern terminus of the Inyo Mountains (the westernmost range regionally); Owens Lake is adjacent to the west with the Sierra Nevada on the other side of the lake. Malpais Mesa is a poorly studied area that has important implications for the development of the Basin and Range Province (Stone et al., 2009). Several geologic areas of interest nearby are: Conglomerate Mesa to the N, Darwin Plateau to the E, and the Coso Volcanic Field to the SW (See Map B in Fig. 14, Stone et al., 2009).



(A)



(B)

Fig 14: Location of Malpais Mesa, Inyo County, CA. (A) Malpais Mesa is situated in the SW portion of the Basin and Range Province, directly east of the Southern Sierra Nevada. (B) Location of Malpais Mesa (green star) in relation to locations of interest in the area such as Death Valley National Park (to the east) and Mount Whitney (to the west). Images courtesy of Google Earth with imagery from the Landsat and Copernicus collections.

Tectonic Setting

Tectonically, Malpais Mesa is bordered to the west by the regionally developed, but mostly unstudied, Inyo Mountain Fault Zone, that cuts through alluvial deposits of Quaternary and Tertiary age (Slemmons et al., 2008). Neighboring Owens Valley is tectonically situated within several zones: the Eastern California seismic zone, Walker Lane (containing the Sierra Nevada Frontal fault zone), and the Eastern California shear zone (Slemmons et.al, 2008). The Eastern California shear zone is transtensional in nature and thought to accommodate 10-13mm/year (20-25% of the total) of the relative dextral motion of the Pacific and North American plates (Bacon et al., 2005). Locally featured is the Eastern Sierran Thrust System (ESTS), a 150 km long contractional belt hypothesized to be caused by buttressing of the Sierran arc. The ESTS is thought to have accommodated 9.3km of shortening throughout its 140-188 Ma deformation history. In the Southern Inyo Mountains, the ESTS produced several arc-parallel shear zones seen in outcrop (Dunne and Walker, 2004).

Mining

Malpais Mesa was an appropriate setting in which to test the StraboSpot mobile app due to the heterogeneity of its geology. Surficially, recent maps categorize most of the area as "Cenozoic Volcanics" (Stone, 2009) but through field investigation, the area also contains sedimentary and metamorphic rocks of Paleozoic and Mesozoic age which are ore-bearing. There are two major lead-silver-zinc deposits which were mined in the area: Talc City Hills and the Santa Rosa Mine. The Talc City Hills at the southern portion of Malpais Mesa hosts ore deposits within the Paleozoic limestone, dolostone, and quartzite which are defined by steeply dipping faults and shear zones. The Santa Rosa Mine is a horst featuring three sets of

mineralized faults and is composed of calc-hornfels and pyroclastics that has been intruded by Cenozoic andesitic and basaltic dikes (Hall and MacKevett, 1958).

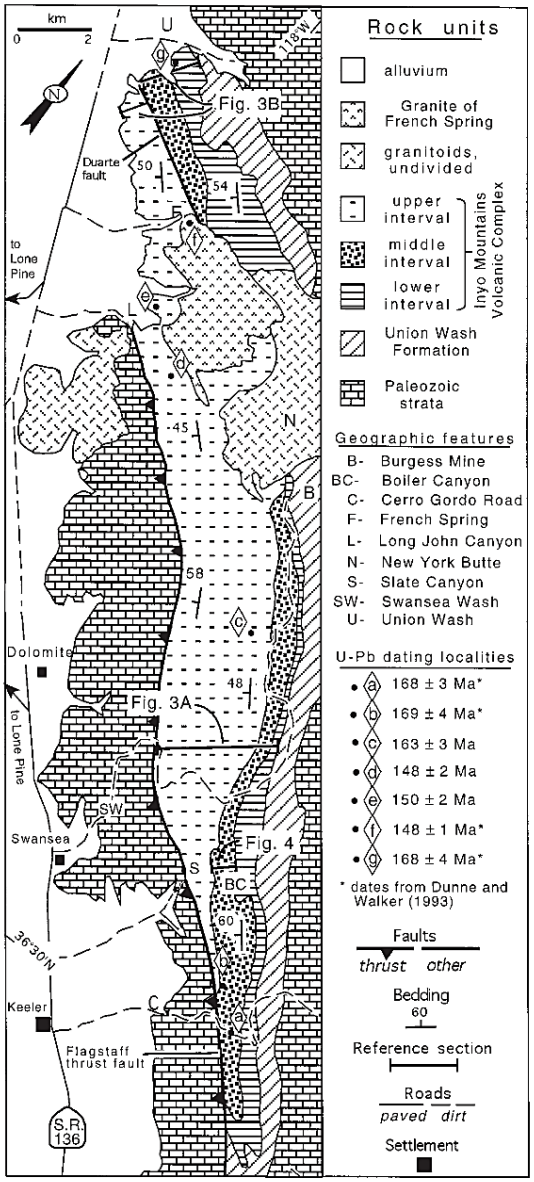


Figure 15: Map of the Southern Inyo Mountains where Malpais Mesa is situated showing the Inyo Mountains Volcanic Complex and surrounding features. Taken from Dunne and others, 1998.

Volcanic Rocks

Malpais Mesa is situated in the southern portion of the Jurassic-age Inyo Mountains Volcanic Complex (IMVC) as first described by Dunne and others, 1998 (Fig. 15). The "core" of Malpais Mesa consists of the IMVC which thins (in map view) to the S and is flanked to the E by Triassic strata and Paleozoic strata to the W, separated from one another by the Flagstaff Thrust Fault of the ESTS (Dunne and Walker, 2004). Elements of the upper, middle, and lower portions of the IMVC can be found around Malpais Mesa. The upper portion consists of volcanogenic clastics, rare calcareous strata, welded tuff, and lava flows. The middle portion contains welded ash-flow tuff, andesitic and rhyolitic lava flows, and some volcanogenic clastics. The lower portion consists of volcanogenic clastics, some basaltic lava flows, and rare felsic tuff; the basal contact consists of a conglomeratic layer then the top of the Union Wash Formation (Stone et al., 2009). The Cenozoic volcanic rocks of the area appear to be

Miocene in age. The neighboring Darwin Plateau (to the SE of Malpais Mesa) features Miocene and Pliocene basaltic flows and pyroclastics. Through K-Ar dating and magnetic polarity, the basaltic flows were found to be mostly 5.3-5.7 Ma with older flows dating to 7-8 Ma corresponding to basin and range extension in the area. Rainbow Canyon, which is just SE of Darwin Plateau shows evidence of cinder cones and basaltic dikes (Schweig III, 1989). Conglomerate Mesa to the immediate N of Malpais Mesa contains basaltic flows, dikes, and pyroclastics also of early Pliocene to latest Miocene in age determined through K-Ar dating (Stone et al., 2009).

Chapter III: Methods

GIS Connection Structure

The ArcGIS Add-In and QGIS Plug-In are designed to download data from the StraboSpot database and upload edited datasets back to StraboSpot. Download and Upload are the two basic functions a user will perform using the Add-In, so the Add-In's toolbar design is divided this way. The Download button on the ArcMap toolbar opens the Download dialog box where the user inputs the information necessary to interact with the StraboSpot database to pull their data down into ArcMap. The Upload button opens the Upload dialog box where edited data are added to StraboSpot.

To launch the QGIS Plug-In, the user clicks a button with the StraboSpot logo on the main QGIS toolbar. This opens a dialog box which guides the user through either a download or upload in much the same way as the ArcGIS Add-In. Differences between the two GIS connections will be detailed in the pseudocode.

Software

Microsoft's Visual Studio Community 2013 was the software program used to design and code the ArcGIS 10.3.1 Add-In, which was written in the VB.NET language. The program conveniently organizes and auto-generates some of the code for the dialog boxes once tools are added. It had a robust integrated development environment (IDE) with the code completion tool, IntelliSense which reduces the need for the programmer to memorize code structure through autocompletion suggestions. Libraries from Microsoft, ArcObjects, Ionic, and Newtonsoft (JSON.NET) were utilized.

JetBrains PyCharm Community Edition 2017 was the software program used to write the Python code for the QGIS Plug-In. Qt Designer with QGIS 2.18.9 custom widgets was the software used to design the GUI. Important Python libraries used include: PyQt4, qgis, osgeo, pyspatialite, requests, json, shutil, pyexif, and psycopg2. Links to view and download the code on GitHub can be found in Appendix 1.

Pseudocode

The following pseudocode describes the steps taken by the StraboSpot/GIS connection products during both types of interactions a user's computer (client) may have with the StraboSpot database (server): download and upload. Differences between the QGIS Plug-In and ArcMap Add-In are noted and are predominantly attributed to differing standards for data storage for each software and differences in the API of each software. The "*" symbol indicates terminology used in StraboSpot's API. See https://strabospot.org/api for REST request examples.

Download Protocol

1.) Authenticate User* (Log-In)

a. Collect email address and password from user.
b. Send credentials to the StraboSpot server through RESTful request.
c. StraboSpot returns “true” or “false” based on whether a user account exists with the given credentials
    i. If “true” the user is advanced to choosing project/dataset(s)
    ii. If “false” the user is asked to re-enter their credentials
d. These credentials must be confirmed as correct to proceed with other StraboSpot interactions. They are sent as part of each following REST request.

2.) User chooses a StraboSpot Project and Dataset(s) to download
    a. Get My Projects* request returns a list of the user’s projects. The user chooses one of these from a list box of names.
    b. Get My Datasets* request returns a list of datasets within the selected project. The user selects all datasets desired for download.

3.) User browses to the folder directory where all download items may be created. Potential items include:
    a. A folder holding the JSON/GeoJSON files created during download as well as any images downloaded from the dataset.
    b. ArcMap Add-In: An ArcGIS File Geodatabase containing the feature classes representing the selected StraboSpot datasets.
    c. QGIS Plug-In: SpatiaLite database containing tables representing the selected StraboSpot datasets from the project.

4.) Create databases:
    a. ArcMap Add-In

i. ArcGIS File Geodatabase (Corresponds to StraboSpot Project level)

ii. Named as "StraboSpotProjectName_mm_dd_yyyy.gdb"

b. QGIS Plug-In (database creation is optional)

i. SpatiaLite database

ii. PostgreSQL database with PostGIS extension using the user specified username, password, and port assuming a local host set up.

5.) For each selected dataset do the following:

a. ArcMap Add-In:

i. Create a feature dataset (Corresponds to the StraboSpot Dataset level) in the file geodatabase. Feature Classes generated in the steps below will be stored there.

ii. For each possible geometry in the dataset (“Point,” “Line,” Polygon”) do the following:

1. A string representing a GeoJSON Feature Collection (written ESRI-style) is started. This later will be written to a file.
2. Get All Fields for Given Dataset* request returns a list of fields which have values in the dataset. The list of fields is inserted into the feature dataset.
3. Get All Spots for Given Dataset* request, using an ‘arc’ variation in the URI, returns all Spots in GeoJSON format. The data are parsed and separated to accommodate ESRI-JSON’s structure. Above all, one feature/row may not contain multiple values for the same field.

4. If directed by the user, images are downloaded and geotagged (optional).
5. The resulting GeoJSON Feature Collection is saved and used as an input in the 'JSON to Feature Class' ArcGIS tool.

iii. Using the data from Get Project* request, create Tags feature classes. These can be linked to the feature classes in the feature dataset through table joins using the "SpotID" field.

b. QGIS Plug-In:

i. In a similar fashion to the ArcMap Add-In steps above, save the entire dataset (Get All Spots for Given Dataset* without 'arc' variation), subsets of the dataset defined by geometry type, and the data from Get Project* (tags are incorporated into the dataset GeoJSON).

1. After parsing the response from each geometry subset of the dataset the resulting GeoJSON is converted to a QGIS Vector Layer, allowing it to appear in the Layers Panel (analogous to the Table of Contents in ArcMap). If the user specified that the data should be saved to either a SpatiaLite or PostgreSQL database, specialized functions are called to save the new layer to a table in the desired database. If no database is specified, the user can save their QGIS Vector Layers in a QGIS Project document, so the current session may be closed.

c. If images were downloaded, a photo layer is added for both applications. In ArcMap, the GeoTagged Photos to Points tool is used and in QGIS HTML is

written for the layer which connects each point with a file location in its table row to the image saved at that location. Both result in the image being displayed as a thumbnail image in a pop-up box (see Fig. 18) if the point is clicked on.

Upload Protocol

1.) Authenticate User*
2.) User chooses an upload method: overwrite, create new, or shapefile uploader (shapefile only available in the ArcMap Add-In). The first two options only work if data has been previously downloaded from StraboSpot.
3.) User chooses the ArcGIS Table of Contents (TOC) or QGIS Layers Panel Layer(s) to upload. These are listed in a list box in the upload dialog box.
4.) If Overwriting chosen, for each selected TOC layer:
    a. Gather identifying information about the data from the Feature Class.
    b. Get All Spots for Given Dataset* request to get the current state of the dataset in StraboSpot.
    c. Run Feature Class to JSON ArcGIS tool on the selected Feature Class or perform QgsVectorFileWriter.writeAsVectorFormat method in QGIS to get each layer as GeoJSON and save.
    d. Signal to StraboSpot to begin versioning procedures on this dataset:
        i. Get Project* request and update the project's modified timestamp to reflect the current date/time. POST the new JSON to StraboSpot through REST request (Update Project*).
            1. If the selected TOC layer represents Tags, the output from getting the JSON will be compared to the project JSON, and any edits will

be identified and applied to the project's Tags in the POST. Code will advance to the next selected TOC layer.

ii. Get Dataset* request and update the dataset's modified timestamp to reflect the current date/time. POST the new JSON to StraboSpot through REST request (Update Dataset*).

e. Compare GeoJSON response from the most recent state of the dataset to the JSON output from c. by utilizing ids to identify and apply any changes from the Feature Class to the GeoJSON response. Save this edited GeoJSON to file.

f. POST the edited GeoJSON to StraboSpot with Upload Features* request.

5.) If Create New Dataset is chosen, for each selected TOC layer:

a. Perform a.-c. from step 4.

b. Get Project* and Update Project* (signal versioning as previously described)

c. Create Dataset* in StraboSpot and Add Dataset to Project* from b.

d. Compare GeoJSON response from Get All Spots From Given Dataset* (the current state of the dataset) to the JSON output from running the Features to JSON tool or running the QgsVectorFileWriter.writeAsVectorFormat method by utilizing ids to identify and apply any changes from the Feature Class to the GeoJSON response. Give all new features in the GeoJSON brand new ids to avoid conflicts in the StraboSpot database. Save this edited GeoJSON to file.

e. POST the edited GeoJSON to populate the new dataset created in c. (Upload Features*).

6.) If Shapefile Uploader tool is chosen in ArcMap, for each selected TOC layer:

a. Gather identifying information about the data from the Feature Class.

b. Create a temporary folder for zipping the shapefiles.

c. Project the shapefile to the WGS84 coordinate system if needed.

d. Find all the associated files for the Shapefile and copy to the temporary folder.

Once all Shapefiles are copied to the temp folder, the folder is zipped and uploaded to StraboSpot's shapefile uploader page.

Chapter IV: Results

The products of running the download protocol outlined in the methods section in ArcGIS yields the following: file geodatabase (fgdb) corresponding to the StraboSpot Project, feature dataset(s) in that fgdb corresponding to the StraboSpot Dataset(s) picked to be downloaded. Within each feature dataset are feature classes containing the data from the dataset. This fgdb is stored in a folder location the user specifies which can also contain JSON files needed for downloading and folders containing images (.jpeg or .tiff format) from each dataset chosen. A StraboSpot Project, which stores multiple datasets, is loosely equivalent in structure to an ArcGIS File Geodatabase and a StraboSpot Dataset, when divided by geometries: point, line, polygon, is equivalent to up to three ArcGIS Feature Classes (Fig. 16). In QGIS, since multiple databases can be used which are more lightweight in design than an ArcGIS fgdb, the feature dataset analogy is lost, but the rest remains the same. Comparison of the GIS and StraboSpot data structures reveals the challenges one faces when translating between these two schemas as discussed below.

Part of the flexibility of StraboSpot datasets, because they are stored as GeoJSON, is that there is no need to separate points, lines, and polygons; however, in a GIS, feature classes are defined by containing data that is of one geometry type (an example can be found in Fig. 16).

Thus, when downloading a StraboSpot Dataset, the GIS connection must make separate RESTful calls for each possible geometry type. Additionally, since tags are stored at the project level, a table of all tags used in the dataset (organized by SpotID) is created and mapped by performing a query using the SpotID.

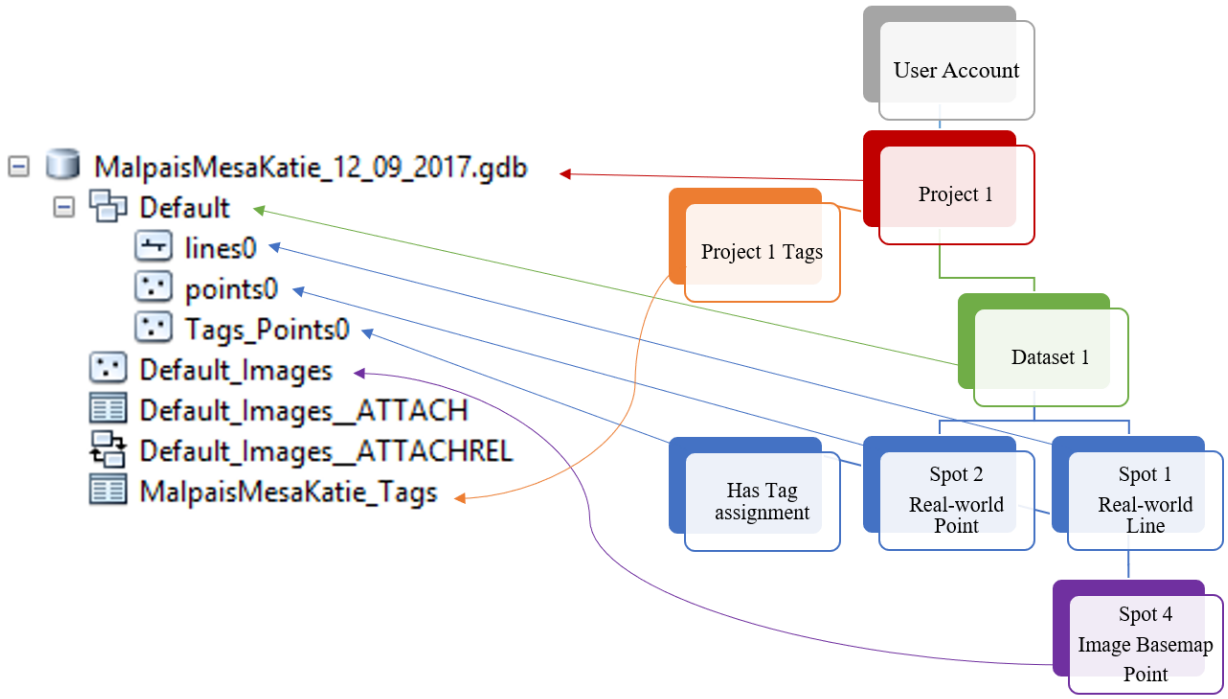


Fig 16: GIS organization schema for data originating from different parts of a StraboSpot project. A project corresponds to a file geodatabase (MalpaisMesaKatie_12_09_2017.gdb), dataset (Default) to a feature dataset, and various types of data to feature classes.

The GeoJSON from each dataset call is parsed into ESRI-style JSON (this also occurs in QGIS) in order to unpack the nested arrays. When a nested array is encountered, a new feature is created. Rows of the same Spot can be linked by SpotID (see Fig. 17).

```
"id": 14758680934587,
"notes": "Goes from a dip of semi horizontal to semi verticals in 10 meter
almost wraps back around",

      "orientation_data": [{
            "type": "planar_orientation",
            "id": 14758682322373,
            "strike": 313.36,
            "dip_direction": 43.36,
            "dip": 32.07,
            "label": "West side of fold",
            "feature_type": "bedding"
      },
      {     "type": "planar_orientation",
            "id": 14758683727799,
            "label": "Top east side of fold",
            "strike": 315.81,
            "dip_direction": 45.81,
            "dip": 79.26,
            "feature_type": "bedding"
      },
      {     "type": "planar_orientation",
            "id": 14758684327981,
            "strike": 183.45,
            "dip_direction": 273.45,
            "dip": 13.24,
            "label": "Bottom west side of fold",
            "feature_type": "bedding"
      }],
```


strike	dip_direction	dip	feature_type	label	notes	SpotID	FeatID
Null	Null	Null	Null	Null	Goes from a dip of semi horizontal to semi verticals in 10 meter almost wraps back around	14758680934587	14758680934587
313.36	43.36	32.07	Bedding	West Side of Fold	Null	14758680934587	14758682322373
315.81	45.81	79.26	Bedding	Top east side of fold	Null	14758680934587	14758683727799
183.45	273.45	13.24	Bedding	Bottom west side of fold	Null	14758680934587	14758684327981

Fig. 17: Example of a nested snippet of a Spot's GeoJSON (top) translated to relational database structure (table). A similar table would be produced during the organization of GeoJSON to ESRI JSON or QGIS GeoJSON. Note that portions of the Spot (individual orientation measurements) can be linked back to the original Spot's 'id' (highlighted in aqua) through the SpotID field, but also have unique feature ids captured in the FeatID field.

Chapter V: Discussion

Challenges

To translate a StraboSpot project containing multiple datasets into a geodatabase containing tables, geometry types must be separated since a geodatabase table can only contain data from one geometry type (point, line, polygon). When downloading a StraboSpot dataset containing data spanning multiple geometry types the number of feature classes produced is equal to the number of unique geometry types of the dataset. To keep the database structure similar to StraboSpot in ArcGIS, an ArcGIS File Geodatabase is analogous to a StraboSpot Project, a Feature Dataset within that file geodatabase is analogous to a StraboSpot Dataset within a Project. The Feature Dataset contains all the data in a single StraboSpot dataset separated by geometry as feature classes. Since Tags are stored at the project level in StraboSpot, they are stored at the file geodatabase level, however, in order to make Tags mappable (since the Project JSON does not contain spatial coordinates) immediately after a user has downloaded a dataset, they are linked to the geometry of the Spot(s) they are used in. This does introduce some redundancy into the database which could be eliminated with the introduction of automatic joins between a Tag table and the dataset feature classes (see future research).

The complications associated with translating between graph and relational database structures are further evidenced in the differences in organization of one piece of data in each system. Data collection in the StraboSpot system is based on the concept of a “Spot” (a node in the graph database) which defines the spatial area over which a set of observations can be applied. A Spot can be a point, line, or polygon “station,” or a “nest” of Spots with similar concepts based on geographic location. Since the StraboSpot database can accept any data, a

Spot is not restricted in the same way a row of data is in a relational database. Traditionally, in a geologic geodatabase, each row in a feature class is equivalent to a "station" where observations are recorded, which will loosely be compared to a Spot, since each row stores a unique set of geographic coordinates. A row is limited to one value per key which here is a field in the feature class attribute table. Additional information for a station could be added using unique IDs and/or information joined across several tables. Unlike a row, a Spot can contain several unique occurrences of the same observation type since the data is organized using GeoJSON. For example, a geologist might choose to record and store multiple strike and dip measurements along a fold in one Spot as seen in Figure 5. This difference between a row and Spot necessitates the "flattening" of a Spot when it needs to become a data entry in a feature class, potentially resulting in one Spot spanning several rows that contain different StraboSpot features (see Fig. 17). Those rows map as multiple geometric symbols on top of each other (e.g. four points occupying the same spatial extent) in ArcMap. A Spot's complete information can be reconstructed by looking at the "SpotID" field in the attribute table. All rows associated with a spot will have the same id in this field, a Unix timestamp in milliseconds plus a random digit, assigned to the Spot by the app. In this way, the deconstruction of a Spot into several rows mirrors the deconstruction of a StraboSpot dataset into several feature classes and illustrates the rigid schema of a relational database compared to a graph database.

The final type of data which needed to be downloaded and incorporated into the geodatabase structure was images. In order to make this happen, each image can be downloaded to the user's computer and saved to the database in a photo layer that is displayed similarly to the photo layer in Google Earth using HTML in each GIS (an example of how this looks in ArcMap is shown in Figure 18). However, the nature of a GIS does not allow for Spots on image

basemaps to be displayed on the image in a GIS as they are in StraboSpot. GIS software uses real-world coordinate systems, while an image basemap in StraboSpot plots points, lines, and polygons in pixel coordinates. All data contained within a Spot placed on an image basemap (e.g. samples, orientation measurements, etc.) is added to a feature class and has the XY coordinates of the next Spot up the hierarchy which contains the image and has real-world coordinates. If a user wanted to plot the image basemap Spots onto an image, they could read the pixel coordinates from the GeoJSON file and plot the Spot using an image processing software program like Illustrator® or Photoshop®.

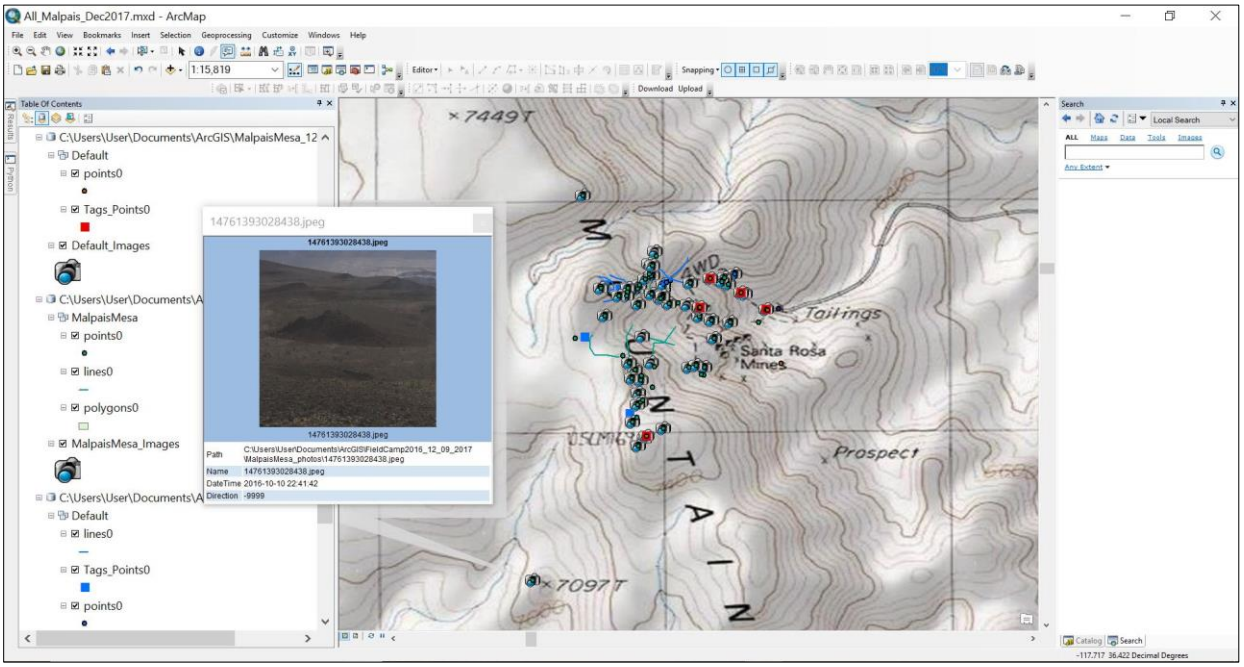


Figure 18: View of all three students' Malpais Mesa StraboSpot Projects downloaded from StraboSpot to ArcMap 10.3.1. The StraboSpot Add-In (used in ArcGIS) is a toolbar tool – the 'Download' and 'Upload' buttons pictured here open dialog boxes which allow StraboSpot users to download data from StraboSpot or upload data modified in ArcMap to StraboSpot. The base layer is ESRI's "USA Topo" basemap, credits: Copyright: © 2013 National Geographic Society, i-cubed.

Considerations for Future Research

The StraboSpot/GIS connections presented in this paper will require dynamic, continued work due to the following factors: new versions of GIS software, the addition of specialized StraboSpot workflows, and bugs/issues reported from further testing.

Within the duration of this project, beginning in Aug. 2015, ArcGIS for Desktop has released three version updates (10.3.1-10.6). ArcMap Add-Ins are version specific, meaning with each version release, the code must be reconfigured to reflect any updates to the ArcGIS for Desktop SDK API and hardcoded to work with the new version. The code update must take place on a computer where the new version is installed, meaning any changes made to a previous version require an additional computer or a roll-back to the previous version which can be time consuming. The ArcGIS Pro API showed numerous changes due to the implementation of Task Asynchronous Programming (TAP) which allows for multithreaded processing and the elimination of DDL operations, so the Add-In will need to be reconfigured with these changes in mind. QGIS Plug-Ins do not require such frequent reconfiguration, however, QGIS 3.x was released in February 2018. The new version of QGIS utilizes Python 3 and QT5, so the Plug-In will need to be updated to reflect these changes to the QGIS Python API.

Currently, StraboSpot is working with the Sedimentology, Petrology and Microstructural geoscience communities to incorporate their lexicons and develop specialized workflows in the app to meet their data collection needs. As StraboSpot's lexicon expands, the GeoJSON parsing component of the GIS connections will require updating to maintain the complete download of datasets. Methods of downloading StraboSpot data in an offline environment using the GeoJSON saved to a user's device when they use the 'Export Project to Device' option within the app have also been explored.

The GIS connections for StraboSpot were released for beta testing in Spring 2018, meaning at the writing of this thesis the number of bugs and issues reported are limited. Throughout the production of the GIS connections, bugs and issues were fixed through rigorous testing utilizing StraboSpot data from field camp users, 'dummy' datasets created to test specific code additions, and other StraboSpot team members. However, the GIS connections have been tested sparsely on other computers, especially those using other operating systems than Windows (such as iOS, Ubuntu, etc.). In conclusion, as the StraboSpot system matures and GIS software release new updates, these GIS connections will need to be revisited and updated.

Chapter VI: Conclusions

Linking StraboSpot with GIS

In recent years, there has been a change in the digital revolution which has been slowly gaining in popularity within the geosciences to incorporate new ideas, namely a demand for more collaboration and sharing of data, and new technology into data collection methods. Tools for data collection in the field remained relatively unchanged in the geosciences until the adoption of portable devices loaded with GIS programs in the late 1990s and the advent of civilian GPS in the early 2000s. There have been numerous efforts to create a standardized GIS database schema which accommodates geoscientists from field collection to completed map products, but all these efforts face challenges inherent with RDBMS schema. Such challenges include: developing a schema which is useful to the individual in the field but can also later be added to larger, community databases with appropriate metadata needed for effective collaboration; building the schema to accommodate data at multiple scales throughout the project

from regional field observations to thin-section geochemical analysis; and the ability to organize data from one field location that may have multiple observations (values) per field (key).

Today's geoscientist is searching for user-friendly, open source methods of collecting, storing, and sharing their data from start to finish. Recent advances in geospatial data designed for use on the Web like GeoJSON and the widespread use of mobile applications along with greater collaboration with computer scientists has led to the creation of novel geologic data management systems. The StraboSpot System, for the collection, storage, and sharing of geologic data is one such creation. StraboSpot consists of a mobile application, available for iOS and Android devices, an online graph database, and now features interoperability with ArcGIS and QGIS. Since StraboSpot's database is graphical instead of relational, translational considerations at both the database and table levels needed to be addressed so that users could have full use of their StraboSpot data once downloaded into a GIS. To design and program GIS connections between StraboSpot and ArcGIS and QGIS required a thorough understanding of JSON/GeoJSON, HTTP and RESTful communication, and the differences between graph and relational databases. The mobile app was beta tested during a field excursion in Fall 2016 to Malpais Mesa, Inyo County, CA to investigate the area's role within the larger Eastern California Shear Zone and produce robust StraboSpot datasets to aid in the development of the GIS connections.

Looking Forward

There are numerous advantages which accompany the further development of digital geologic data collection systems such as StraboSpot. The StraboSpot developers aim to design workflows and interfaces on the app, online database website, and GIS connections which are intuitive to use and require little knowledge of the technologies powering the system. Through additional collaboration with other geoscientific communities and interoperability with other

databases and apps like Rick Allmendinger's StereonetMobile, StraboSpot will become a more complete all-in-one data system for geoscientists. Just as GIS led the geosciences out of the "paper map problem," so will new workflows and systems lead to: greater ease of collaboration through open-source data sharing, more cost-effective science since data is more likely to be reused, an increase in reproducibility as well as preservation of historic geologic occurrences leading to better understanding of scientific phenomena, and more public engagement with scientific findings (Fowler, 2016). A common phrase in GIS is, "If a picture is worth a thousand words, then a map must be worth a thousand pictures," but the development of digital methods of geologic data collection has exposed a wealth of data behind a geologic map waiting to be harnessed for future research.

References

Amirian, P., 2013, Beginning ArcGIS for Desktop Development using. NET: John Wiley & Sons.

Bacon, S. N., Jayko, A. S., and McGeehin, J. P., 2005, Holocene and latest Pleistocene oblique dextral faulting on the southern Inyo Mountains fault, Owens Lake Basin, California: Bull. Seismol. Soc. Am., 95, 2472–2485, doi:10.1785/0120040228.

Bostock, M., and Davies, J, 2013, Code as cartography: The Cartographic Journal 50.2: 129-135.

Butler, H., Daly, M., Doyle, A., Gillies, S., Hagen, S., Schaub, T., 2016, The GeoJSON Format: No. RFC 7946.

Chasseur, C., Li, Y., and Patel, J.M., 2013, Enabling JSON Document Stores in Relational Systems: WebDB. Vol. 13.

DiScala, M., and Abadi, D.J., 2016, Automatic generation of normalized relational schemas from nested key-value data: Proceedings of the 2016 International Conference on Management of Data. ACM.

Dunne, G. C., Garvey, T. P., Oborne, M., Schneidereit, D., Fritsche, A. E., and Walker, J. D., 1998, Geology of the Inyo Mountains Volcanic Complex: Implications for Jurassic paleogeography of the Sierran magmatic arc in eastern California: Geol. Soc. Am. Bull., 110, 1376– 1397.

Dunne, G.C., and Walker, J.D., 2004, Structure and evolution of the East Sierran thrust system, east central California: Tectonics 23.4.

ESRI, 2010, How to Run a Geoprocessing Tool: ArcObjects 10 .NET SDK Help, help.arcgis.com/en/sdk/10.0/arcobjects_net/conceptualhelp/index.html.

Feng, X., Shen, J., and Fan, Y., 2009, REST: An alternative to RPC for Web services architecture: Future Information Networks, ICFIN 2009, First International Conference on IEEE, 2009.

Fielding, R. T., and Taylor, R.N., 2000, Architectural styles and the design of network-based software architectures: Doctoral dissertation: University of California, Irvine.

Fowler, R., 2016, Embracing open data in field-driven sciences: Eos, 97, doi:10.1029/2016EO047789. Published on 10 March 2016.

Hall, W.E., and MacKevett, E.M., 1958, Economic geology of the Darwin Quadrangle, Inyo County, California: Vol. 51. 1958.

Jones, R.R., McCaffrey, K.J.W., Wilson, R.W., Holdsworth, R.E., 2004, Digital field data acquisition: towards increased quantification of uncertainty during geological mapping: Geological Society, London, Special Publications 239.1: 43-56.

Loudon, T.V., 2000, Geoscience after IT: a view of the present and future impact of information technology on geoscience: Vol. 17. Elsevier.

McCaffrey, K. J. W., Jones, R.R., Holdsworth, R.E., Wilson, R.W., Clegg, P., Imber, J., Holliman, N., Trinks, I., 2005, Unlocking the spatial dimension: digital technologies and the future of geoscience fieldwork: Journal of the Geological Society 162.6: 927-938.

Mookerjee, M., Vieira, D., Chan, M.A., Gil, Y., Pavlis, T.L., Spear, F.S., and Tikoff, B., 2015, Field data management: Integrating cyberscience and geoscience, Eos, 96, doi:10.1029/2015EO036703. Published on 13 October 2015

Pavlis, T.L., et al. "Computer-based data acquisition and visualization systems in field geology: Results from 12 years of experimentation and future potential." Geosphere 6.3 (2010): 275-294.

Redmond, E., and Wilson, J.R., 2012, Seven databases in seven weeks: *The Pragmatic Bookshelf*.

Robinson, I., Webber, J., and Eifrem, E., 2015, Graph databases: new opportunities for connected data: O'Reilly Media, Inc.

Rufledt, C., Walker, J.D., et al., 2017, StraboSpot Help Guide: https://strabospot.org/files/Strabo_Help_Guide.pdf.

Schweig III, E.S., 1989, Basin-range tectonics in the Darwin Plateau, southwestern Great Basin, California: Geological Society of America Bulletin 101.5: 652-662.

Slemmons, D.B., Vittori, E., Jayko, A.S., Carver, G.A., Bacon, S.N., 2008, Quaternary Fault and Lineament Map of Owens Valley, Inyo County, Eastern California: Map and Chart Series 96. Geological Society of America.

Stone P., Swanson B.J., Stevens C.H., Dunne G.C., Priest S.S., 2009, Geologic Map of the Southern Inyo Mountains and Vicinity, Inyo County, California: U.S. Geological Survey Scientific Investigations Map 3094, 1:24,000

Tomlinson, R.F., 1974, The Application of Electronic Computing Methods and Techniques to the Storage, Compilation and Assessment of Mapped Data: Diss. University College London (University of London).

Turner, A., 2014, ArcGIS Online Supports GeoJSON: ESRI, ArcGIS Blog. Published 16 Dec. 2014. Web Accessed 26 June 2017.

USGS National Cooperative Geologic Mapping Program (NCGMP), 2018, GeMS (Geologic Map Schema) -- a standard format for digital publication of geologic maps: Draft manuscript, to be published after revision as USGS Techniques & Methods publication; accessed May 2018 at https://ngmdb.usgs.gov/Info/standards/GeMS/docs/GeMSv2_draft7g_ProvisionalRelease.pdf; see also https://ngmdb.usgs.gov/Info/standards/GeMS/.

Walker, J.D., Black, R.A., 2000, Mapping the Outcrop: Geotimes vol.45, no. 11, p.28-31.

Walker, J.D., Black, R.A., Linn, J.K., Thomas, A.J., Wiseman, R., D'Attilio, M.G., 1996, Development of geographic information systems-oriented databases for integrated geological and geophysical applications: GSA Today v. 6: 1-6.

Newman, J., Tikoff, B., Walker, J.D., Good, J., Michels, Z.D., Ash, J., Andrew, J., Williams, R.T., Richard, S.M., 2015, Development of Structural Geology and Tectonics Data System with Field and Lab Interface, presented at 2015, Fall Meeting, AGU, San Francisco, CA, 14-18 December

Walker, J.D., Tikoff, B., Newman, J., Clark, R., Ash, J.M., Good, J., Bunse, E.G., Möller, A., Kahn, M., Williams, R., Michels, Z., Andrew, J.E., and Rufledt, C., submitted, StraboSpot data system for Structural Geology: Geosphere.

Whitmeyer, S., Feely, M., De Paor, D., Hennessy, R., Whitmeyer, S., Nicoletti, J., Santangelo, B., Daniels, J., Rivera, M., 2009, Visualization techniques in field geology education: A case study from western Ireland: Field geology education: Historical perspectives and modern approaches v. 461: 105. DOI: https://doi.org/10.1130/2009.2461(10)

Wilde, E., 2007, Putting things to REST: School of Information, UC Berkeley.

Vicknair, C., Macias, M., Zhao, Z., Nan, X., Chen, Y., Wilkings, D., 2010, A Comparison of a Graph Database and a Relational Database: ACM SE '10 Proceedings of the 48th Annual Southeast Regional Conference Article No. 42.

Zhang, X., Song, W., and Liu, L., 2014, An implementation approach to store GIS spatial data on NoSQL database: Geoinformatics (GeoInformatics), 2014 22nd International Conference on IEEE.

Appendix A: Links

All code associated with the StraboSpot project can be found and viewed on GitHub. The organization's page is: https://github.com/StraboSpot. The GIS connections can be found in repositories on this page:

The StraboSpot/ArcGIS Add-In code can be found in the "strabo-arcgis-plugin" repository: https://github.com/StraboSpot/strabo-arcgis-plugin.

The StraboSpot/QGIS Plug-In code can be found in the "strabo-qgis-plugin" repository: https://github.com/StraboSpot/strabo-qgis-plugin.

Follow the directions in the README file on each page to learn how to use the GIS connections.

Videos detailing how to use StraboSpot can be found on our "StraboSpot" YouTube page: https://www.youtube.com/channel/UC-bWoOvRvpedOswFERXQPiw.