Scalable Image Informatics

Dmitry Fedorov, B. S. Manjunath, Christian A. Lang,
Kristian Kvilekval
Center for Multimodal Big Data Science and Healthcare
University of California, Santa Barbara

August 14, 2016

1 Introduction

Images and video play a major role in scientific discoveries. Significant new advances in imaging science over the past two decades have resulted in new devices and technologies that are able to probe the world at nanoscales to planetary scales. These instruments generate massive amounts of multimodal imaging data. In addition to the raw imaging data, these instruments capture additional critical information – the metadata– that include the imaging context. Further, the experimental conditions are often added manually to such metadata that describe processes that are not implicit in the instrumentation metadata. Despite these technological advances in imaging sciences, resources for curation, distribution, sharing and analysis of such data at scale, are still lacking. Robust image analysis workflows have the potential to transform image-based sciences such as biology, ecology, remote sensing, materials science and medical imaging. In this context, this Chapter presents BisQue, a novel eco-system where scientific image analysis methods can be discovered, tested, verified, refined and shared amongst users on a shared, cloud based infrastructure. The vision of BisQue is to enable large-scale, data driven scientific explorations. The following sections will discuss the core requirements of such an architecture, challenges in developing and deploying the methods, and will conclude with an application to image recognition using deep learning.

1.1 Core Requirements

The development of BisQue is driven by the requirements of the scientific imaging community. Figure 1 summarizes the core requirements that supports ubiquitous access to multidimensional (5-D) images and other binary data types.

- Heterogeneous data: Typical scientific datasets are heterogeneous in nature (e.g., multi-dimensional 3D volumes together with 1-D sequence information). A comprehensive solution to managing such data is critical for discovery and innovation.

- Metadata integration: Contextual information in most scientific imaging experiments are embedded in the associated metadata that describe instrument-specific and experiment specific parameters.
- Large scale data management: Modern experiments are typically statistical in nature and thus require analyzing large datasets.
- Integrated data analysis: Further, most experiments carry carefully curated manual annotations and manipulations that are critical for further computations.
- Integration with automated analytics: Equally critical is that the analysis modules work closely with the data and metadata, and the results of such analysis is integrated into the overall system for data mining.
- Provenance tracking: Integration of the data and methods will allow provenance tracking towards enabling reproducible computations.
- Support for large-scale data analytics: This is critical for high-throughput imaging applications that routinely generate large amounts of data.
- Support for easy integration of analysis modules: Most new experiments are designed with new computational analysis in mind and thus require support for easy integration of new and complex analysis routines.
- Support for indexing, searching and querying high-dimensional and graph data. This is important as many commonly used multimodal features are high-dimensional vectors and organizing such data is a major challenge for pattern recognition and data mining methods.
- Developer support: In addition to the scientific users, an extensible system should provide support for researchers outside of a chosen scientific domain who would be able to contribute novel analytics and benefit from testing/validating their methods on a diverse data. This is critical for scientific knowledge discovery.

Towards addressing these above challenges, the BisQue image informatics system has pioneered extensible image analysis on the web with full web standards compliance and large data support. BisQue introduced the schema-less hierarchical and flexible data model for annotations so as to address the needs of diverse labs/users which has proven most useful for storing heterogeneous data and meta-data. The flexible annotations are a key element allowing rapid integration of image analysis and its results into the system. BisQue is designed from the ground-up to be scalable and deployable on cloud computing infrastructure and BisQue can be easily deployed using Docker virtualization [8]. Deep learning and data analytics architectures are being integrated with BisQue, thus harnessing the collective power of data and methods in order to derive new insights.

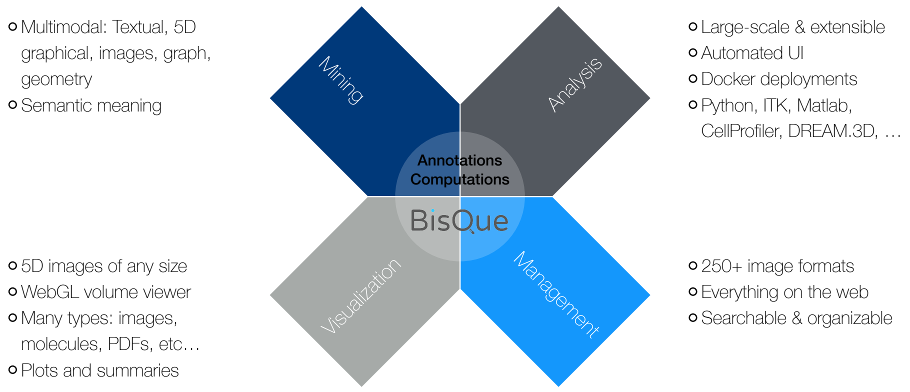

Figure 1: The development of BisQue is driven by the requirements of the scientific imaging community. At the core is the data/feature infrastructure that supports metadata resources and ubiquitous access to multidimensional (5-D) images and other binary data types.

2 Core Concepts

At its core, the BisQue system describes data objects (e.g., images, tables, experiments) *via* trees of user-defined tuples of metadata, where each element in the tree may be described with a user given name, value, type and units. There are additional attributes controlled by the system enabling ownership and access control, time of creation and access, ontological reference, etc. Annotation tuples can simply be textual annotations as well as more complex graphical annotations described *via* several multi-dimensional graphical primitives, such as points, polylines, polygons, surfaces, and more complex shapes.

The metadata trees are called documents or resources and may describe 0, 1, or many binary files accessible to the system. Each resource and tuples contained within are addressable *via* Universal Resource Locators (URIs) allowing the creation of links to sub-elements in each document. Types of these resources and tuples suggest specific micro-services that can operate on referenced binary resources and provide domain-specific operations.

All metadata documents are handled by the data service which is used to orchestrate the query system and any other binary services. Other resource types include an image resource, serviceable *via* an image service. Images can be composed of one or multiple files and the metadata elements of the image resource may describe the geometry and physical characteristics of these files composing a multidimensional image. Furthermore, dense table data (such as HDF5) can be serviced by a table service allowing slicing and dicing. Chemical data (such as molecule SD files) can be serviced by a chemistry service providing typical visualization and queries performed over molecular data.

The metadata resources are used not only to describe binary files but also every other concept used in the system, such as users or module executions. Internal and external APIs, micro-service communication and historical differ-

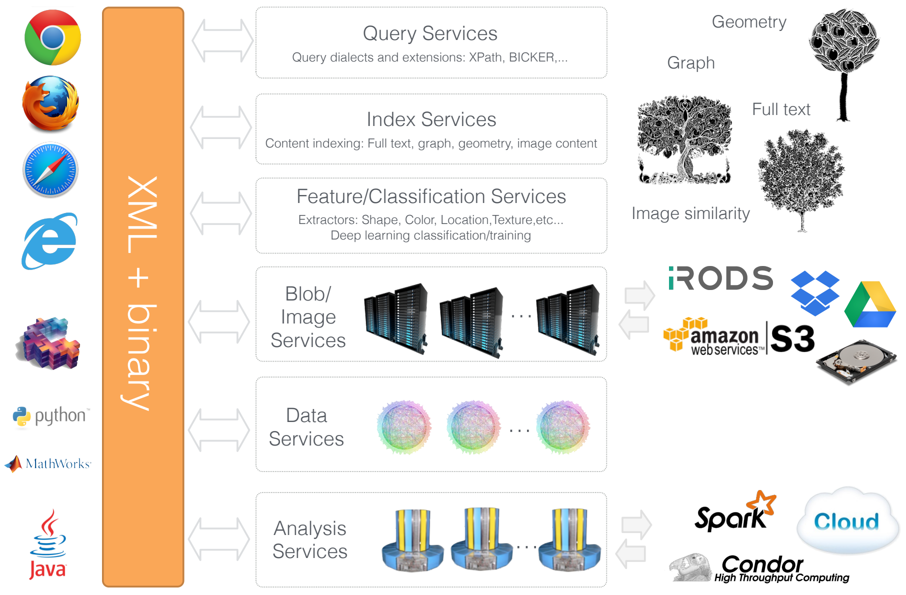
XML + binary
Query Services
Query dialects and extensions: XPath, BICKER,...
Index Services
Content indexing: Full text, graph, geometry, image content
Feature/Classification Services
Extractors: Shape, Color, Location,Texture,etc...
Deep learning classification/training
Blob/ Image Services
Data Services
Analysis Services
Geometry
Graph
Full text
Image similarity
iRODS
amazon web services S3
Spark
Cloud
Condor High Throughput Computing
python
MathWorks
Java

Figure 2: Bisque is implemented as a scalable and modular web-service. Image Servers store and manipulate images. Data servers provide flexible metadata storage. Extension servers house executable modules. The client server seamlessly integrates services across storage and processing hardware. Communication between various components is performed in a RESTful manner through HTTP requests carrying XML and JSON. Backend servers seamlessly provide access to local resources as well as large scalable resources like Amazon S3 or iRODS for data storage or Condor grids for computation.

ences are also described with resource documents. The canonical representation of the resource document is in XML due to the well described xpath/xquery interfaces that allow slicing and dicing complex hierarchical documents. Services can also represent metadata trees in other formats such as JSON or CSV when exchanging uniform data in a more efficient manner. The dense data elements can be described in the canonical XML representation (for exchange or indexing) but presenting many dense elements in such a form may be unwieldy. Thus binary data formats are also used for exchange by specialized micro-services, such as images or HDF-5.

The BisQue system orchestrates multiple micro-services providing access control for asynchronous operations, users and user contributed analyses (Fig 2). Services are tightly integrated with the system and have access to internal data structures and scalability. User contributed analyses, called modules, execute in a sand-boxed environment and only possess temporary user-level credentials while executing. The module system on the other hand automatically scales module executions based on available resources and module requirements.

Sharing and collaboration are extremely important for any scientific endeavor and the web nature of the BisQue system makes that easy. Flexible data format support by the micro services optimizes data transfers for editing and visualization. Proximity to computational resources enables large scale computations. In addition, the system keeps a complete record of all resultant data, analysis code that produced it and users who ran it, guaranteeing strict provenance and reproducibility of all of its data computed collaboratively.

2.1 Metadata graph

The hierarchical and flexible annotations enabled by BisQue allows user and developers to rapidly contribute data, add annotations and integrate analysis modules. Metadata trees can represent virtually any user-defined schema. Semantic meaning for elements in the tree can be achieved by a simple user-defined type or more rigorously by pointing to an ontology definition provided by some service. Pointers (URIs) within documents are used extensively in order to connect different resources. For example, an image may point to a microscope definition document as well as a sample preparation document. Such links form metadata graphs, allowing aggregated graph queries and also providing access to specific binary micro-services.

This rich metadata representation is too expensive to represent very large amounts of uniform (dense) data (millions or billions of elements) thus pointers can be used to simply link a branch in the metadata tree to a dense data element. Such elements are typically served by micro-services representing table data (e.g., detected cells with computed features) or pixel data (e.g., image data or segmented masks).

2.2 Versioning, Provenance and Queries

The provenance of scientific data is also important for multi-user environments (such as laboratories and companies) in order to ensure the validity and reproducibility of scientific discoveries. At the same time the nature of scientific process is based on experimentation and thus involves a lot of trial and error. The BisQue system solves these concerns with versioned metadata documents and strict system-controlled analysis provenance that both encourages experimentation and guarantees strict provenance of analysis results.

The metadata system preserves all the changes to the metadata by storing validity time intervals for each data item. This information allows the recreation of any document at any point in time, historical queries and the representation of changes *via* delta documents.

All analysis executions are described by a system-controlled Module Execution Document (MEX) that contains pointers to an exact module being used for computation, all input elements, such as pointers to input resources (e.g., images, tables) and explicit module parameters. They also contain all the produced outputs either explicitly or by linking. In addition, module documents identify the exact state of the source code or binary algorithm at module execution time by additionally storing a source code repository reference.

Document versioning enables user correction of automatically produced results and provides an exact change that could be useful for improvement of automated methods. This system thus ensures strict provenance of every produced resultant data element and encourages experimentation. Additionally, provenance paired with strict versioning of analysis code guarantees reproducibility of all the computed results.

The BisQue system encourages storing of different types of data needed to describe an experiment. In order to derive scientific insights from these heterogeneous data a query system able to handle multi-modal data is required. BisQue's unified query system uses a SPARQL-like query language that provides an abstraction from the underlying data stores and indices and allows expression of complex queries over graphs of metadata documents. Depending on the availability of indices and the amount of data queried, the query system can decide to execute a query by pushing it down into an underlying relational database (e.g., PostgreSQL) or it may decide to run it *via* brute-force computation in a distributed fashion (e.g., aggregating millions of data items *via* Apache Spark). These two approaches require very different computational infrastructure and algorithmic support.

The BisQue system allows each micro-service responsible for its data-type to contribute one or more indices/summarizers for the unified query system based on expandable functions. The query system will in turn dispatch queries to specific sub-indices and later aggregate them into one single answer. The data service keeps track of all the available indices and sends them updates whenever a document changes. Each index decides whether the update is relevant to their indexable data-type and whether to update its indices instantaneously or queue them for a later batch update. Such indices/summaries cover as varied data types as full text, metadata graph, ontological terms, graphical annotations geometry, image-based similarity, molecular scaffold similarity, and statistical summaries of numerical dense data.

For more complex data analytics requirements that can not be easily expressed with BisQue's query language or that are provided as program code (e.g., in Matlab, Python or C), BisQue's analysis module system can be utilized. BisQue modules can be parallelized on a computational cluster in a Map-Reduce or in a more complex Directed Acyclic Graph (DAG) fashion in order to maximize parallelism. Each execution can in turn utilize any scalable micro-service and their indices. The orchestration of the distributed module execution, data/code shipment, and result collection is handled transparently by the module system.

3 Basic micro-services

BisQue is designed following common web techniques and benefits from the available hardware and software infrastructure. Cloud infrastructure, RESTful

APIs, light-weight virtual machines, HTTP scalability and caching are all used by BisQue. Another design motif utilized throughout the system is lazy evaluation paired with extensive caching as it offers fine grained control of dispatch and reduces overall computations by skipping unneeded work.

3.1 Uniform metadata representation and query orchestration: data service

The main service within the BisQue system is the data service responsible for storage, access and query orchestration over the metadata documents. The data service may restrict access to certain elements of system-defined documents such as user descriptions or module execution documents. These schema restrictions and enforcements are provided by type specific micro-services and are orchestrated by the data service. User defined or system-unknown types will not provoke any specialized behaviour or restrictions and thus allow natural system extensions. Extensible, type-driven user interface elements are described later in the appropriate section. The data service is also responsible for converting metadata documents to/from XML, JSON and other formats.

The most important orchestration function of the data service is in multi-modal queries. The data service provides XPath-like and SPARQL-like ("BQR") query languages supporting extended functionality by accessing multiple indexers *via* plug-in functions. While both languages allow slicing and dicing of large hierarchical documents, XPath is more expressive for tree-structured queries (e.g., "get all descendant nodes of this node") while BQR is more expressive for graph-structured queries (e.g., "find nodes in documents linked by this document"). In fact, the XPath query processor builds on the BQR query processor, whereby reducing code complexity.

The data service notifies all the known indexers about any changes to a metadata document. Indexers make a decision on whether to immediately update themselves or queue their update in an asynchronous fashion. Some indexers may require resource-intensive computations of feature descriptors and thus asynchronous updates would be preferred. Each indexer maintains their own data structures needed to efficiently answer specific queries (e.g., an R-tree index structure for low-dimensional nearest neighbor queries). The BQR language provides primitives to query specific indexes and on linking them with the other parts of the query (e.g., "find documents linking cells that are very similar in shape to this cell").

3.2 Scalability of micro-services and analysis

There are multiple scalability approaches required by the BisQue components. Services utilize three basic scalability mechanisms. The first one scales micro-services themselves in order to support a large number of concurrent users. This technique utilizes standard web technologies for load detection and request distribution. Cloud technologies allow automated scale-up and down of service machines based on current user load.

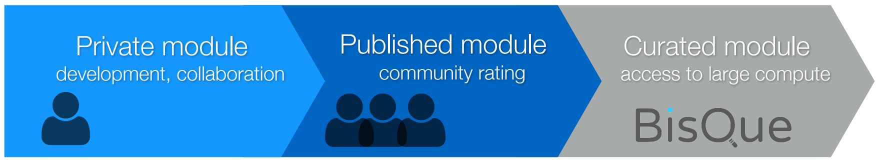

Figure 3: The BisQue module development life-cycle.

The second mechanism allows services to off-load a slow operation when a response time-out is exhausted over to the background asynchronous processing system. It is required due to some operations being computationally expensive (e.g., complex data analytics queries). In this case a service will respond with a document indicating the operation is still in progress and provide a return URL where a response can eventually be obtained. The requester can then periodically poll to check if the result is ready.

The third mechanism is specifically designed for user-contributed analysis and utilizes a slower but massively scalable cluster dispatch mechanism. Compute jobs (i.e., modules) are scheduled to one or more cluster nodes based on their hardware and software requirements. This is discussed in more detail in Section 4.3.

3.3 Analysis extensions: module service

The module service enables users to easily contribute their own analyses (e.g., image classification and recognition methods) and allows dispatching these analyses to the available computational infrastructure (ee Section 4 for more details.) The module service operates on two system-defined resource types:

1. module description documents that define formal inputs and outputs of the algorithm, user interfaces, source code location and version along with other descriptors;

2. module execution documents that describe a particular execution, values of its inputs and outputs, execution status and initial and final date and time.

Since modules are described by metadata documents they can be shared with collaborators as well as published for community participation, these modules can later be curated by the system administrator as demonstrated on Figure 3.

Module execution is initialized by passing a document templated from module definition and containing required input values. A developer can permit automatic data-parallel execution by simply defining an iterable input parameter. A more complex execution can be achieved by passing a DAG document composed of multiple module execution templates.

Automatically parallelized execution is only allowed in batch mode where all the required inputs are defined *a priori*. Interactive analysis on the other hand has only partial inputs initially and will request additional user inputs *via* the user interface while running until the end of the execution is reached. Such a module may remain executing and waiting for input parameters for a prolonged period of time.

Additionally, the module service provides mechanisms to monitor currently executing modules and it facilitates communication between the user interface and its running module code.

3.4 Uniform representation of heterogeneous storage subsystems: blob service

Modern institutions already utilize large storage systems, often they manage multiple different systems and often those systems are read-only. Moreover, individual users also use several storage services. To further complicate matters, data storage is available at different speeds with different price points. It may be advantageous to store old data in a cheaper but slower system while keeping most current data elements in a much faster but more expensive system.

BisQue allows bringing disparate storage mechanisms together and enabling annotation and analysis of data stored within those systems. The blob service handles multiple storage systems and their authentication *via* extensible drivers that can handle large local file systems and enterprise solutions like iRODS, Amazon S3, Box, Google Drive and others. The blob service also handles local caching of these resources for improved multiple access performance.

Metadata documents describing files located in remote storage systems simply store URIs that define a driver, a specific attached store and a path to the file. This allows annotating large amounts of resources without ever moving any bytes until those bytes are required for visualization or analysis.

Another benefit of this descriptory mechanism is with cold storage. Elements located or moved to cold storage can be rapidly found using metadata descriptors without access to bytes. Moreover, caching of derivative results may enable fast partial visualization, for example, image service may provide pixel preview without accessing original data by caching derivative thumbnails.

3.5 Uniform access and operations over data files: image service and table service

Accessing bytes does not yet allow accessing functional information since scientific data is typically stored in a multitude of proprietary formats. Providing uniform access to those formats is another goal of a truly interoperable system.

Due to their size, an important goal is to keep the data as close as possible to the computational infrastructure used to analyze and visualize the data. For example, an image viewer can only show a number of pixels available on the screen which is usually a tiny fraction of pixels available in the scientific image itself. At the same time, the visualization process typically needs to access a much larger number of pixels in order to compute the required view of the data. Similarly, visualization of a numeric table with a billion rows only needs to show a few hundred rows at a time. Thus the bandwidth required for preparation of the view is much larger than the view itself allowing remote viewers. Analysis of these data, on the other hand, typically requires access to most of the bytes and thus faster access is desired.

BisQue utilizes micro-services responsible for a specific logical data type. These services reside in data centers on fast hardware with very good network connectivity and typically on the same local network with the institutional data storage (e.g., CyVerse Atmosphere+iRODS or Amazon EC2+S3). Each service is able to handle multiple formats while providing a single uniform API offering all the functionality needed for visualization and analysis. In the following, we describe in more detail services that provide access to the most basic data types used in image informatics workflows.

3.5.1 Image service

The image service provides access to the most important data type in image informatics: images. It offers support for more than 250 image formats by combining most widely used decoding and encoding libraries: our own C++ libbioimage, OpenSlide, Imaris Convert, BioFormats, FFmpeg, GDCM, and others. This allows BisQue to support a wide gamut of life-sciences imaging modalities from 5D fluorescence, large EM connectomics data, behavioral video data, underwater imagery, GIS aerial and satellite imagery, medical imaging CT/MRI/ultrasound to histopathology whole slide imaging.

There is a large number of typical image processing operations that can be requested in any sequence on input data. They include slicing and dicing, extracting tiles and resolutions, transformations of colors, bit depths, geometric and spatial, intensity projections, interpolations, fusions, histogram operations and many others. All these operations are considered views of original image data and never modify the original pixels.

Another very important function of the image service is to extract and present in a uniform manner metadata embedded in image files. Most modern microscopes embed a large amount of acquisition and instrument parameters that can be useful for data interpretation and processing.

Many other services use image service for image data, such as feature service, image content indexing, classification and others.

3.5.2 Table service

Another common data type used in bio-imaging is the dense numeric table. They are commonly used to store features extracted from images, like cells with many measured parameters, numeric descriptors and other data. Table service provides uniform access to most common formats like CSV, HDF-5, Excel as well as other services like Paradigm4 SciDB. Typical operations provided *via* the RESTful API are slicing and dicing and various transformations. Large numbers of graphical annotations can also be stored in these dense formats and later used for further computations or visualized at varying scale-levels.

4 Analysis modules

Facilitating analysis within a database framework was the original BisQue motivation, and BisQue is unique in terms of providing such an integrated storage, analysis and visualization environment. There are many disparate analysis packages widely used in life-sciences. Bringing them all together in a simple integrated manner along with custom analysis routines proves very useful to create custom workflows. The BisQue system makes it easy to rapidly script custom analysis and run at a large scale, it also allows creating complex and custom user interfaces shareable with collaborators. In the following we will describe ways of bringing custom and already available analysis modules to the system.

4.1 Python and Matlab scripting

Python has gained large popularity in the scientific community and offers a large number of high quality libraries from native SciPy to easy-to-use python bindings for libraries such as the ITK [6]. Matlab has been a popular language for a while and there are many libraries available to its users. Bringing these under the same umbrella may save a lot of development and validation time. BisQue offers an elegant way to bring these analyses from experimentation to large scale execution.

Any user can request a special authentication token for local analysis which can then be used for an interactive remote session in any of the programming languages. BisQue offers APIs for both Python and Matlab that simplify most aspects of communication with the system. Interactive session permits a developer to explore the system while using small data portions. Finalized scripts can easily be converted to BisQue modules and used for large scale processing.

4.2 Pipeline support

Instead of using low-level programming language logic, many scientific analysis tools rely on higher-level "execution pipelines" to describe the specific logic in which analysis steps are performed and what data flows from one step to the next. BisQue allows importing pipelines as first-order resources that can be annotated, searched, and viewed. Like with other supported resources, BisQue preserves the original pipeline files. Examples of supported pipelines include Dream.3D for materials science [5] and CellProfiler for biology [2].

Imported pipelines can contain placeholders for parameters to be filled in by the user when starting the module execution. The module user interface renders each of the placeholders as a run parameter and instantiates the pipeline accordingly before execution starts. As with other BisQue modules, these input parameters are preserved for later inspection as part of the module execution documents.

4.3 Complex module execution descriptors

The BisQue module execution document allows the specification of directed acyclic execution graphs where each step is run only when the nodes of all incoming edges have completed execution. A simple example is a graph with a pre-run phase to perform some pre-processing, followed by five parallel runs of the main analysis phase, followed by a post-run phase to collect and summarize results. Computer-based analysis of scientific data oftentimes requires analysis steps to be run with many input parameter combinations in order to understand the effect on the generated output. For this purpose, BisQue's module execution system allows to specify parameter ranges and automatically executes the module instances in parallel, one for each parameter combination.

Depending on the underlying compute cluster system and the module execution graph, BisQue may either orchestrate the execution itself or may push it down to the cluster scheduler. An example for the latter is the execution of a DAG of tasks by the Condor task scheduler. In either case, the system has to ensure to schedule tasks only to nodes that are consistent with the task's requirements (e.g., sufficient main memory, availability of GPUs). The use of Docker containerization technology allows the utilization of heterogeneous clusters, without the need to configure or pre-install software on each node.

Since the parallel instances may update the BisQue metadata store at any time, care has to be taken to ensure that the data remains consistent without sacrificing parallelism. For example, the XML document describing the module run may contain many subsections (one for each parameter combination). Since they are independent of each other, they can be updated in parallel without locking the entire document. For this purpose, the module execution relies heavily on the proper concurrency control of the underlying data service.

5 Building on the concepts: sparse images

Image mosaics (montages) are very popular with microscopists to augment the instrument's field of view. They have become a requirement for studies in connectomics by generating images of at very large scales. Microscopes equipped with automated stages and embedded vibrotomes can produce volumetric images iteratively slicing the tissue and providing an approximate location of an image in the physical volume. Light-sheet microscopes paired with modern clearing techniques enable scientists to image and study whole organs. In all of these cases, microscopes produce a large number of images possibly with their approximate locations in 3D space.

The typical workflow is to undergo a heavy processing by refining the location of each block within the volume using automatic image registration and subsequently produce a large dense image by geometrically transforming each input block into the output discrete volume before the whole image can be visualized and analyzed. The desire to keep the original images typically means doubling the storage requirement as well as a difficulty to immediately identify which portion of the final volume do these images correspond to.

In contrast, BisQue utilizes a more dynamic and lazy approach by describing a "sparse" image composed of references to original images along with associated geometric transformations. Such a structure can represent a very complex multi-dimensional image. It can be rapidly imported and visualized in the system. Each image block in this construct can be independently transformed by scalable image services while the client-side image viewer decides what portions to present on the screen.

Moreover, "sparse" images can grow and change over time as, for example, additional data blocks are acquired by the microscope. Thus allowing visualizing large mosaics as they are being acquired and/or geometrically refined. BisQue supports this by using a "mosaic" metadata document and a specialized service called Montager which operates on a mosaic type to refine transformations, generate fused overlapping pixels for visualization and eventually generate a large densified image in a background asynchronous process. Montager uses image service extensively to create transformed derivatives of input image blocks and the feature service to compute point descriptors for geometric transformation refinements.

6 Feature Services and Machine Leaning

Image recognition often requires basic feature extraction. More recently, deep learning based pattern recognition techniques have demonstrated highly promising results in various computer vision applications. BisQue supports a diversity of feature computations through its feature service, and the next release of BisQue will include integration with deep learning architectures for scalable computations.

6.1 Feature service

The BisQue feature service is responsible for the computation of numeric feature descriptors in a scalable and validated manner. It offers more than 100 commonly used numerical descriptors (HTD, EHD, SCD, DCD, SIFT, SURF, HAR, Wavelet Histogram, Radon Coefficients, Chebishev Statistics, ...) [12] computed on image derivatives (provided by the image service) as well as graphical annotations. In order to guarantee correctness we have integrated, validated, corrected and rewritten code from multiple well know libraries: OpenCV [1], Mahotas [3], MPEG7 [7], WndChrm [10]. This provides a basic building block for any classifier in taking care of the training/testing dataset creation. The same service can later be used at classification time to compute descriptors for data to be classified. The demonstration of this approach is available as "Botanicam" classification and training modules [11].

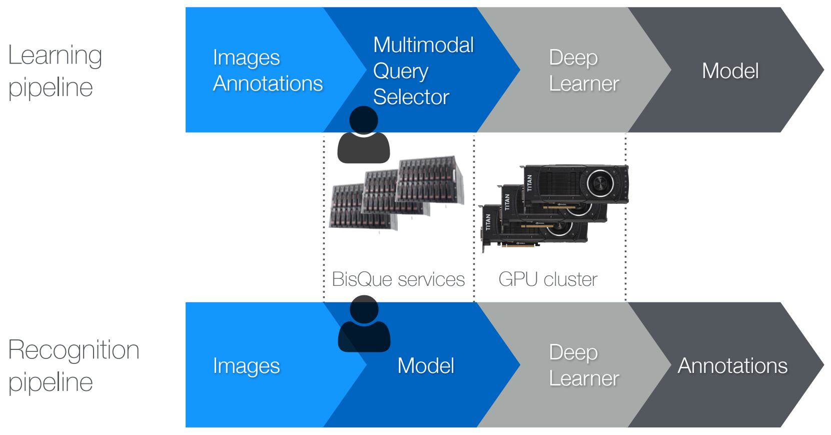

Figure 4: Deep learning training and classification pipelines.

6.2 Connoisseur service for deep learning

The Connoisseur service is an integrated training and classification solution for image recognition based on deep learning. Connoisseur uses Convolutional Neural Networks (CNNs) in order to create a model directly from training data without feature selection. This enables any domain scientist to create a specialized classifier model directly from annotated still or video imagery without the need to know the engineering intricacies of classifier design. Any of the BisQue's organizational and filtering functions to choose the training dataset, and could possibly include all of the available data. Connoisseur presents all classes found within the selected data and then shows how well it did against the automatically chosen testing part (never seen by the training) of the dataset. At this point a scientist can choose to discard certain classes if their performance was below the required level.

During classification, each sample is given a confidence score computed from multiple measurements performed in a vicinity of the location of interest. This measure is adjustable by the user to skip low confidence samples. Automated annotations can be validated and modified by the human expert creating more data. These new samples can then be used to improve the model over time, thus a model is a dynamic object that is constantly updated.

Not surprisingly, one of the more time consuming parts of the training process is the preparation of the training, testing and validation datasets. All image data must be in a specific format and size as defined by the training model which usually means extraction of small patches around annotated regions, possibly resizing those patches, ensuring the same color space and profile as well as pixel depth in bytes. Considering hundreds of annotations per image this operation will be repeated millions of times per typical datasets. This embarrassingly parallel processing is ideally handled by the scalable BisQue image service. Here, multiple asynchronous background Connoisseur processes are requesting the image derivatives following the metadata found in the identified set and writing the training database in parallel.

Once the training database is created, a single multi-GPU server is used to effectively train the model. A typical dataset could be trained in a matter of a few hours on a modern 4-GPU (nVidia TitanX) server using the BisQue asynchronous background processing facility.

A model metadata document is created to describe model files. This includes the classes detected in the identified dataset, numbers of samples per class, and their accuracies and errors once the training is done. This model document will be updated with every consecutive training session.

Once the model is trained, data classification is quite efficient and is embarrassingly parallel. The slowest process here is the initialization of the GPU library and loading of the CNN model into the GPU memory. A typical CNN model's size is about 500MB (depending on the network topology) and therefore multiple models can be loaded at the same time.

6.3 Connoisseur module for domain experts

The Connoisseur classification module offers a user-friendly interface, parallel execution over datasets and permanent storage and provenance of resulting annotations. A user can choose a dataset to classify, a pre-trained model and a classification mode. There are three classification modes offered by Connoisseur.

The first one creates uniformly or randomly distributed point annotations. It is designed to automate the widely used percent cover technique. Each automated point is marked with an accuracy measure and allows visual selection of the desired level.

The second mode is a fast Voronoi partitioning of the image. It is useful for object-environment co-occurrence questions. For example, brittle stars over mud versus rock in underwater images or cancer cells near fat or blood vessels in microscopy data.

While the previous two methods produce graphical annotations (vector data), the third one produces a mask image with a higher quality but slower semantic segmentation where the image is partitioned in classified regions. Each region has an associated accuracy measure and can also be pruned to a desired level.

7 Application Example: Annotation and Classification of Underwater Images

Here we briefly describe an example application to marine sciences. Researchers at the Marine Science Institute at UCSB are using BisQue to store, manage, annotate and develope automated analysis techniques for the Marine Biodiversity Observation Network (MBON) project. Figure 5 shows the BisQue annotation interface configured for percent cover with 100 sample points. The types (species) of annotations (visible in the right side text widget) are user-defined and can grow and change as needed over time. This enables continuous evolution of the annotations to fit the evolving needs of the project.

It is desired to have many annotators due to the number of training images required. A typical dataset may contain about 300 different classes of interest and thus would need hundreds of thousands of training annotations. The BisQue UI allows spitting a dataset among a number of annotators. Further it also facilitates validation and accepting the annotations by an independent expert.

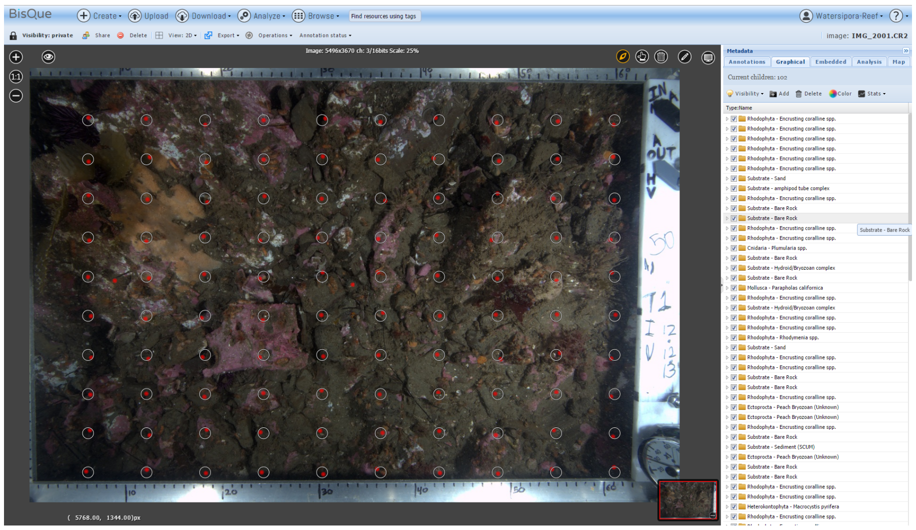
BisQue
Create
Upload
Download
Analyze
Browse
Find resources using tags
Watersipora-Reef
Visibility: private
Share
Delete
View: 2D
Export
Operations
Annotation status
image: IMG_2001.CR2
Image: 5496x3670 ch: 3/16bits Scale: 25%
Metadata
Annotations
Graphical
Embedded
Analysis
Map
Current children: 100
Visibility
Add
Delete
Color
Stats
Type:Name
Rhodophyta - Encrusting coralline spp.
Substrate - Sand
Substrate - amphipod tube complex
Substrate - Bare Rock
Cnidaria - Plumularia spp.
Substrate - Hydroid/Bryozoan complex
Mollusca - Parapholas californica
Rhodophyta - Rhodymenia spp.
Ectoprocta - Peach Bryozoan (Unknown)
Substrate - Sediment (SCUM)
Heterokontophyta - Macrocystis pyrifera
(5768.00, 1344.00)px

Figure 5: Manual annotation interface configured for percent cover with 100 points. The types (species) of annotations (visible in the right side widget) are user-defined and can grow and change as needed over time. This enables continuous evolution of the annotations to fit the evolving needs of the project.

One particular study included a dataset of over 2,000 underwater images and manually annotated for percent coverage of sessile species. Each image contained 100 annotated locations amounting to $> 200K$ data points with over 30 species. Over 80% of these data points are covered by the 11 most abundant classes. We obtained 85% classification accuracy on these 11 classes using two different feature aggregation techniques, one using CRF based models and the other using a K-NN classification with dropout regularization [9]. The Connoisseur based deep learning technique demonstrated an even higher average accuracy of 94.73% with an error of 3.65% on the same dataset [4], demonstrating the power of state-of-the-art CNN approaches.

An example of the uniform percent cover classification automation at 95% confidence is presented in the Figure 6. Different classes are shown in different colors. Figure 7 shows an example result for the third mode of classification that results in segmented regions for the same 95% confidence.

8 Summary

We presented an extensible image informatics platform, BisQue, for reproducible multimodal data analytics. While the initial motivation for BisQue came from the life sciences, these requirements cut across most scientific imaging applications. Some of the recent applications include marine sciences, materials science, medical imaging and health care. BisQue is unique in its integration of multimodal databases with data analytics, making it possible to track the data and its processing, including provenance on the methods themselves. BisQue adopts the state of the art in web based analytics and cloud computing, making it easy for the end users to immediately take advantage of the latest methods. At the same time it enables researchers in computer vision, pattern recognition and machine learning to work with diverse types of data at scale. BisQue is available as a core service through the CyVerse cyber infrastructure (http://cyverse.org) as well as open source for download.

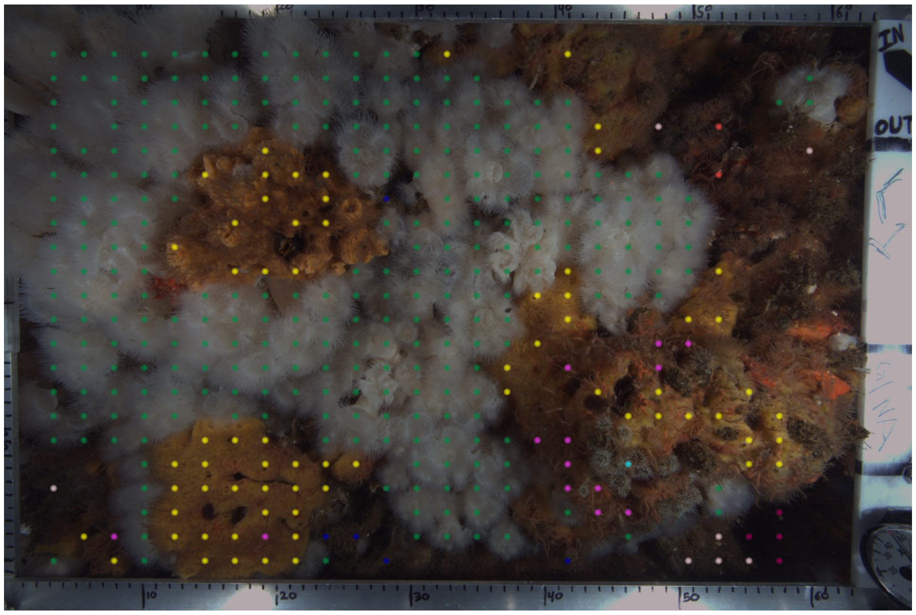

Figure 6: Automated point annotations imitating percent cover annotations at 95% confidence.

References

[1] G. Bradski. Opencv library. Dr. Dobb's Journal of Software Tools, 2000.

[2] A. Carpenter, T. Jones, M. Lamprecht, C. Clarke, I. Kang, O. Friman, D. Guertin, J. Chang, R. Lindquist, J. Moffat, P. Golland, and D. Sabatini. Cellprofiler: image analysis software for identifying and quantifying cell phenotypes. Genome Biology, 7(R:100), Oct. 2006.

[3] L. P. Coelho. Mahotas: Open source software for scriptable computer vision. CoRR, abs/1211.4907, 2012.

[4] D. Fedorov, K. Kvilekval, B. Manjunath, and R. Miller. Bisque: cloud-based system for management, annotation, visualization, analysis and data mining of underwater and remote sensing imagery. In Poster at Ocean Sciences Meeting, New Orleans, LA, Feb 2016.

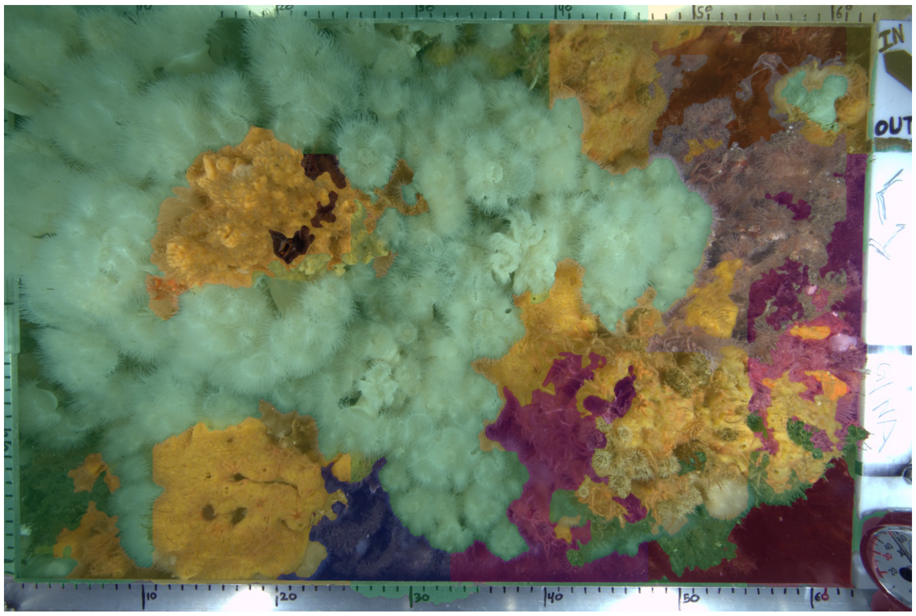

Figure 7: Semantic segmentation of at 95% confidence.

[5] M. A. Groeber and M. A. Jackson. Dream.3d: A digital representation environment for the analysis of microstructure in 3d. Integrating Materials and Manufacturing Innovation, 3(1):1–17, 2014.

[6] B. Lowekamp, D. Chen, L. Ibanez, and D. Blezek. The design of simpleitk. Frontiers in Neuroinformatics, 7:45, 2013.

[7] B. Manjunath, P. Salembier, and T. Sikora. Introduction to MPEG-7: Multimedia Content Description Interface. John Wiley & Sons, Inc., New York, NY, USA, 2002.

[8] D. Merkel. Docker: Lightweight linux containers for consistent development and deployment. Linux J., 2014(239), Mar. 2014.

[9] A. Rahimi, D. Fedorov, S. Sunderrajan, B. Manjunath, R. Miller, B. Doheny, and H. Page. Marine biodiversity classification using dropout regularization. In Workshop on Computer Vision for Analysis of Underwater Imagery: International Conference on Pattern recognition, Stockholm, Sweden, Aug 2014.

[10] L. Shamir, N. Orlov, D. M. Eckley, T. J. Macura, J. Johnston, and I. G. Goldberg. Wndchrm - an open source utility for biological image analysis. Source Code for Biology and Medicine, 3, 2008.

[11] C. Wheat, D. Fedorov, G. Abdollahian, and K. Kvilekval. Botanicam: The plant recognizer, 2008. `http://bisque.ece.ucsb.edu/module_service/Botanicam/`.

[12] C. Wheat, D. Fedorov, and K. Kvilekval. Feature service, 2010. `http://bisque.ece.ucsb.edu/features/list`.