Platform for Spatial Molecular Data

by

Vivek Dasari

S.B., Massachusetts Institute of Technology (2014)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Master of Engineering in Computer Science and Molecular Biology
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June 2015

Author .... Signature redacted .....
Department of Electrical Engineering and Computer Science
May 22, 2015

Certified by. Signature redacted .....
Richard Terry
Lead Senior Staff Scientist
Thesis Supervisor

Certified by.. Signature redacted .....
George Church
Professor of Health Sciences and Technology
Thesis Supervisor

Accepted by .. Signature redacted .....
Albert R. Meyer
Chairman, Department Committee on Graduate Theses

Platform for Spatial Molecular Data

by

Vivek Dasari

Submitted to the Department of Electrical Engineering and Computer Science
on May 22, 2015, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Computer Science and Molecular Biology

Abstract

I designed and implemented a comprehensive platform for storing, analyzing, visualizing, and interacting with spatial molecular data. With the advent of high throughput *in situ* sequencing methods, such as fluorescent *in situ* sequencing (FISSEQ), the need for a platform to organize spatial molecular data has become paramount. The platform is divided into seven services: raw data handling, a spatial coordinate system, an analysis service, an image service, a molecular data service, a spatial data service and a visualization service. Together, these services compose a modular system for organizing the next generation of spatial molecular data.

Thesis Supervisor: Richard Terry
Title: Lead Senior Staff Scientist

Thesis Supervisor: George Church
Title: Professor of Health Sciences and Technology

Acknowledgments

I would like to thank Evan Daugharthy for all of his guidance, contributions, and support in my undergraduate and graduate research. I'd also like to thank my parents, Satish Dasari and Veda Dasari, and my brother, Sidarth Dasari, for their moral support and assistance throughout the process.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the goals behind developing a platform for spatial molecular data. Additionally, it will discuss the motivation behind the project and outline a high level overview of the platform. Chapter two describes spatial molecular data in detail. In addition, existing services and their limitations will be discussed as well. Chapter three comprehensively describes the technical architecture that is used to build the platform for spatial molecular data. Critical design decisions are explained as well. Chapter four enumerates current and future use cases of the platform. Chapter five proposes future enhancements that can build on and improve the existing platform.

1.1 Motivation and Purpose

With the development of Fluorescent *in situ* sequencing (FISSEQ), it has become possible to collect high resolution spatial information about biological systems. This data is rich with information that can be used to develop insights in both academic research and translational medicine. Unfortunately, the tools and techniques required to fully analyze spatial data are only beginning to emerge. Additionally, storing and analyzing spatial data is very resource intensive.

Spatial molecular data is very powerful. By combining traditional molecular data with spatial coordinates, it becomes possible to understand the molecular mechanisms that propel life in greater detail. For example, the FISSEQ technique has been applied

to a fibroblast wound assay to discover novel genes involved in the wound healing pathway. Without the spatial dimension, traditional molecular data is inherently limited to providing information at a population averaged level. Spatial molecular data is not limited in this manner; therefore, insights from spatial molecular data are more accurate and reliable.

The high resolution nature of spatial molecular data makes it difficult to analyze without the aid of software. To fully recognize the potential of spatial molecular data, a suite of software tools must be developed to analyze this novel data. These tools do not exist yet for a number of reasons. First, spatial molecular data is fairly new, so data scientists have not had time to develop and evaluate their tools. Additionally, there is no easy way to access this spatial data. Without quick and easy access, the spatial analytical methods are being developed slowly.

The purpose of this platform is to provide the foundation for a future suite of spatial analytical tools as well as an initial organization of the spatial molecular data. With a well documented interface and modular service structure, this platform will serve as the access point for the spatial molecular data. Ideally, this will also make it easier for spatial analytical method developers to build and evaluate their tools.

1.2 Outline of Services

1.2.1 Goals of Service-oriented Architecture

The platform will be separated into seven separate services; each service has a single responsibility and a documented interface. The service-oriented architecture has a number of benefits. Loose coupling between services improves modularity. Services can be replaced or edited individually as long as the specified interface remains the same. Loose coupling allows the platform be technology and implementation agnostic. Service abstraction hides unnecessary details and keeps the application interfaces clean. The service architecture makes the system easily extensible by simply adding another service. The service architecture is also easily scalable. As the the amount

of spatial molecular data collected grows, the load can be distributed across as many servers as necessary to keep the platform functional.

1.2.2 List of Services

1. Raw images from the sequencing machine are passed directly to the raw data handling service. A basic implementation of the data handling service is responsible for storing the updating the raw image metadata and storing the raw images via the image service.

2. The image service is responsible for managing the images. This includes saving the images to a filesystem, serving the images on a public domain, and processing any request to create, edit, or delete an image in the service.

3. The analysis service is responsible for organizing and processing computationally expensive analytical methods. It can read and write to the spatial data service, molecular data service, and image service. One key analysis service is the object identification pipeline. An object identifier detects spatial data, constructs a mesh representation, and saves the object via the spatial data service.

4. The coordinate system service is a special type of analysis service. The coordinate system service manages the global coordinate system across images and spatial data in the same experimental universe.

5. The spatial data service is responsible for storing spatial data and processing spatial queries. Spatial relationship queries, such as finding all objects contained by another object, are supported.

6. The molecular data service manages the annotations on the images and spatial data. The service also interfaces with other molecular biological databases, such as those hosted by the National Center of Biotechnology Information (NCBI).

7. The visualization service interfaces with all of these services to retrieve the data requested by the user and display it. The visualization service is the only service

to live on the client.

Chapter 2

Background

2.1 Description of Spatial Molecular Data

Living organisms are composed of molecular components. Molecular components include nucleic acids monomers like adenine, guanine, cytosine, thymine and uracil. Individual amino acids can also be described as a molecular component. Large chains of these primary components can be composed to perform useful functions in living organisms. For example, chains of nucleic acid monomers create polymers of DNA and RNA. Long chains of amino acids monomers create proteins that perform functions at the molecular scale. The central dogma of molecular biology describes the process of DNA being transcribed to RNA that are later translated into proteins. Proteins are the basic functional units inside of cells that are essential to life. These building blocks can compose together with other macromolecules, such as lipids, metabolites, biomolecules, and other small molecules, to form larger entities that perform more complex functions. Combinations of these macromolecules build cellular components including: organelles, membranes, nuclei, nucleoli, vacuoles, endoplasmic reticulum, Golgi apparatus, mitochondria, granules, P bodies, microtubules, cilia, and more.

Spatial molecular data is the detection of these molecular components and their precise location in intracellular space and extracellular space. By observing the precise location of where specific components are functioning, it is possible to learn immensely more about molecular biology and the underlying processes involved in life. The

spatial information is particularly important in context of spatial relationships. It is crucial to know how far or close molecular components are from each other in order to make meaningful conclusions. This realization led to the development of a global coordinate system per experiment context (described later).

2.2 Methods of Gathering Spatial Molecular Data

The initial motivation for the development of this platform stemmed from the advancements made in fluorescent *in situ* sequencing (FISSEQ). The FISSEQ protocol can be used to detect molecular substrates *in situ*. The first step involves identity encoding, where information about the identity of a molecular substrate is captured and produces a useful sequencing template that can be detected. The next step involves signal amplification such that it becomes possible to detect with an imaging technique. Signal amplification is followed by the sequencing method that converts the sequencing template into an detectable optical signal. Finally, error correction techniques are used to clean the resultant signal. The resulting output is a signal (image) where spatial molecular data can be extracted and analyzed. Other techniques to extract spatial molecular data include Seurat and MERFISH. It is highly probable that more techniques will be developed in the near future as the power for spatial molecular data is recognized and detection modalities improve.

2.3 Existing Tools

Due to the novelty of spatial molecular data, there is no existing system designed to store, analyze, and visualize spatial molecular data. A system that assumes all of these responsibilities is necessary in order to make actionable insights with spatial molecular data. There are, however, existing technologies that can be leveraged to help build a platform for spatial molecular data.

One particularly useful technology to store spatial data efficiently and perform spatial relationship queries is the postGIS extension for postgreSQL. PostGIS was

designed to help store and query mapping data; however, it has recently been extended to support 3D cartesian space as well as 3D meshes. PostGIS stores spatial data efficiently in a binary format. It would be difficult to use PostGIS alone to make insights from spatial molecular data because it has no visualization component and no image store to analyze images along with the spatial data.

Although there are some tools that can render models in 3D, they are often standalone products that do not interface fluidly with other services. Additionally, none of the tools found could recreate a digital world from image stacks. However, it would be possible to build this functionality with THREE.js. By building the visualization from scratch, it was possible to make a visualization service that interfaces across many services.

Chapter 3

Technical System Description

3.1 Overview

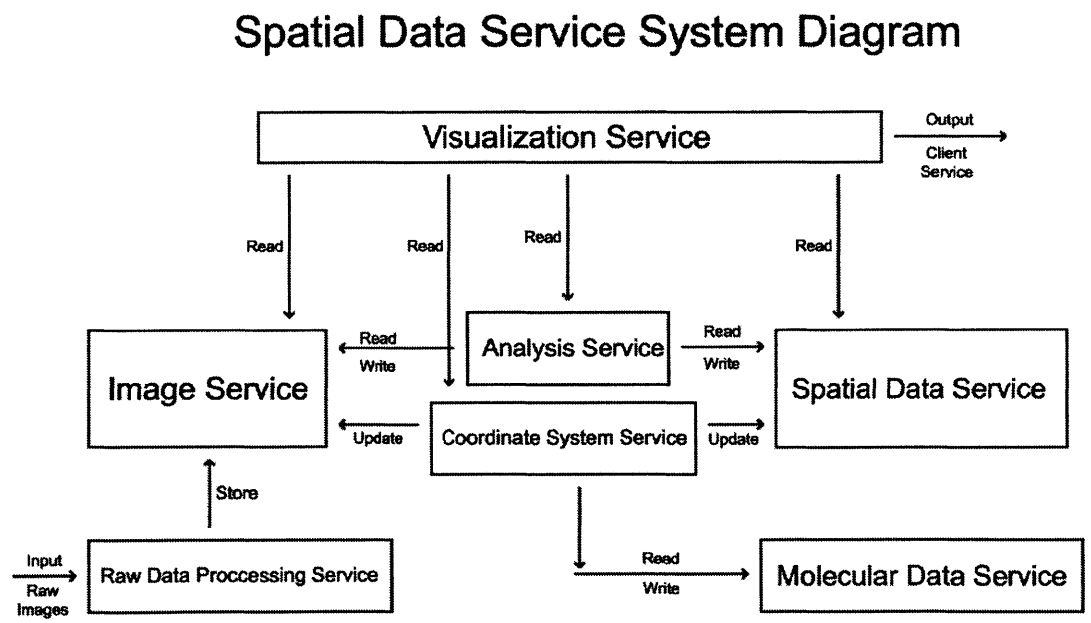

Figure 3-1: Architecture overview. Arrows correspond to public interface between services. Note that the visualization service lives separately on the client. All other services reside on one or more servers.

A system overview is depicted in Figure 3-1. The following will describe the operations performed by the system when a user is interacting with the system. Users

can interact with the platform in two ways: uploading new data or interacting with existing data.

If a user wants to begin using the platform, they must begin by providing the raw data handling service with images containing molecular data. The raw data handling service writes metadata on the images, such as the name of the experiment, the experiment being conducted, the time of processing, and much more. It also the corrects any detected errors in the metadata position attributes written by the imaging machine, so images are properly aligned along the Z-stack. In addition, it provides feedback to the machine about any misalignments, which helps the imaging robot correct for errors as well. The raw data handling service passes the image to the image service with a request to save the image.

The image service saves the image on its local filesystem in a directory that is being publicly hosted. This provides the image a specific uniform resource identifier(URI), and the image becomes available to access through the internet, which is how other services can access the image. The image's public url is be appended to its metadata. Next, the image service indexes the image metadata and saves the metadata in a searchable database. The last of the image service duties is to notify the coordinate system service and object identification pipeline of the new image. Once notified, the coordinate system service updates the global position metadata attributes of the image to place the image in context of the other relevant images. Additionally, each experiment context has its own global coordinate system that relates all of the images and objects in that context. The coordinate system service saves each of the global coordinate systems in its own database for quick lookup. The object identification pipeline varies by implementation; a variety of object identification software packages can be used, such as CellProfiler or Visualization Toolkit. After identifying objects, they are saved by the spatial data service. Once the objects are saved, the user can now interact with the uploaded dataset and all identified objects.

In addition to uploading data, a user can interact with the data through the interface provided by the visualization service. The visualization service provides a graphical user interface (GUI) to visualize and manipulate the images and spatial

data. When the visualization service is initialized, it loads a 3-D space and populates it with a light and a camera. The camera's location is controlled by the user input. The visualization service queries the image service and spatial data service for appropriate images and data based on the position of the camera and a specified zoom level. The user may click on an object representation to interact with it. A full list of interactions in the visualization section below. Depending the on the interaction, the visualization service may interface with the image service, analysis service, spatial data service, and/or the molecular data service to retrieve the requested information. The requested information is displayed back to the user via the visualization service. Users can also choose to interact with the platform by directly accessing the public endpoints on each service.

3.2 Raw Data Service

The raw data service manages images as they arrive from the sequencing machine.

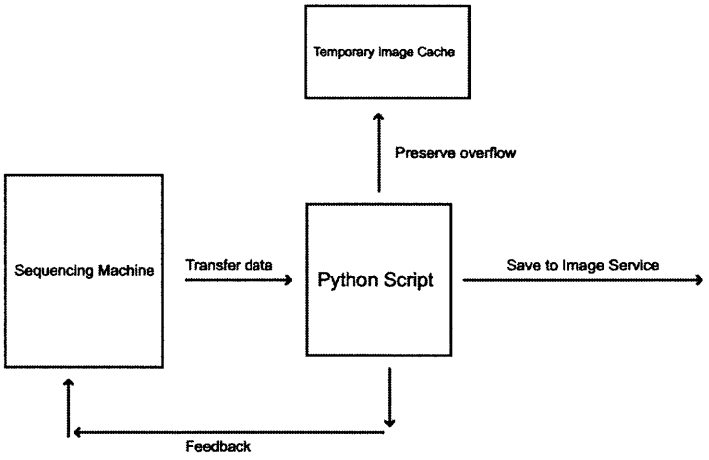

Figure 3-2: System Diagram of the Raw Data Service. The service manages data sent from directly from the sequencing machine. The temporary image cache is used as a buffer to minimize data loss.

A socket is used to stream image pixel values to the raw data service. The raw data service attempts to process the stream immediately; however, a temporary cache

is implemented to store data if the main process is busy or blocked. In addition to pixel values, the sequencing machine is responsible for sending over all experimental and image acquisition metadata. This data includes: experimenter information, brief experiment description, cell type, light sources, filters, detectors, magnification, exposure times, encoder values and any other information necessary to reproduce the image. The raw data service is primarily responsible for ensuring that the position annotations for each image are correct by comparing neighboring pixels in all dimensions. After fixing any issues with the position annotation, the raw data service sends a POST request to the image service to save the image and its corresponding image data.

3.3 Image Service

The image service manages all of the images across the entire platform.

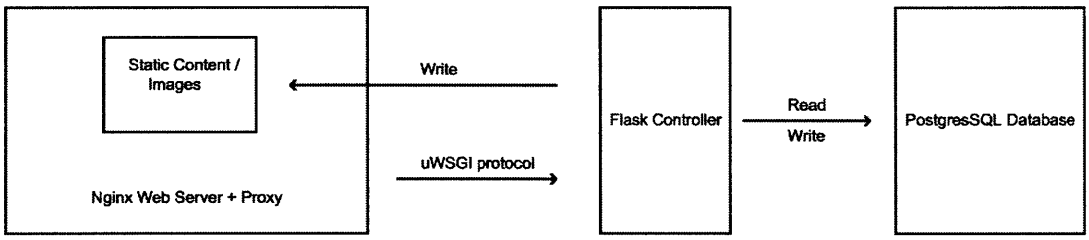

Figure 3-3: Outline of the image service architecture. Ngnix efficiently hosts static files. It also communicates with the Flask controller with the uWSGI protocol. The flask server can write images to the directory of publicly hosted static assets. The controller also puts metadata from the images into the PostgreSQL database. The controller is responsible for maintaining the database.

3.3.1 Implementation

Storing and displaying large quantities of searchable, high resolution images is a difficult challenge. The image service consists of a filesystem, database, controller, and web server. The web server hosts a specified public assets directory, so the directory and its content to be accessed via the web. The Nginx HTTP server and

reverse proxy is used to efficiently host the static content. The controller handles requests to the image service and may spawn asynchronous processes if necessary. When the request is completed, the controller returns a response to the original requester with a status code and any requested data. The controller is written with the flask microframework and exposes a REST API. The GET, POST, PUT, and DELETE HTTP operations can be performed on any image resource. GET is used to retrieve and view images. It is an idempotent operation. POST adds new images to the service. PUT updates an image and DELETE removes an image from the service. All of these operations search for the requested image or images in the database. The database is a postgresSQL implementation that stores metadata about each image. An example schema for TIFF images is depicted below. The key attributes that are indexed for quick lookup include: experimenter, XGlobal, YGlobal, ZGlobal. The URI where the image can be accessed through the Nginx server is stored with the image metadata. For scalability purposes, the image service passes the image URI internally and as a response to a request for an image instead transferring raw image bytes.

In addition to storing raw images, the image service is responsible for mipmapping images, so they are be available to view at multiple zoom levels. The graphics card on the machine running the image service is responsible for generating default mipmaps. Mipmaps are generated by using either trilinear interpolation or anisotropic filtering. Mipmaps can be manually created with image processing software and saved over the automatically generated mipmaps. This is a critical process because the visualization service uses images as textures to build the visualization. Depending on the resolution required from the image, the proper mipmap must be loaded as a texture. Otherwise, a visualization of many images at low resolution does not piece together a coherent image. The raw images are acquired at a resolution of 4096 x 4096. Twelve mipamps can be generated from each image, so the system has twelve discreet zoom levels. The twelve mipmaps only required 33% more storage space than the original image.

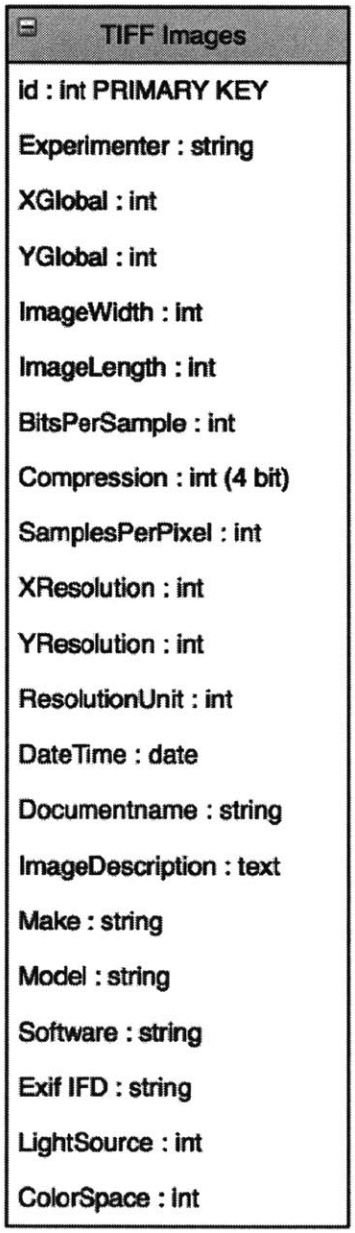

Figure 3-4: An example schema of image metadata stored by the image service's database.

3.3.2 Design Decisions

Nginx was chosen to be the primary proxy and web server primarily for its speed at serving static content, which is the primary function of the image service. Unlike other web servers, like Apache, Nginx cannot natively process dynamic content. This means the overhead of the dynamic interpreter is bypassed when static content is requested on an Nginx server. The overall result is that static content is served quicker. The image service will be serving a lot of static content, so speed is a priority. The uWSGI protocol is used to configure the Flask microframework with the Nginx web server. Additionally, Nginx is ideal for scalability because of its an event-based, asynchronous, and non-blocking architecture.

3.4 Analysis Service

The analysis service is a platform for analytical methods.

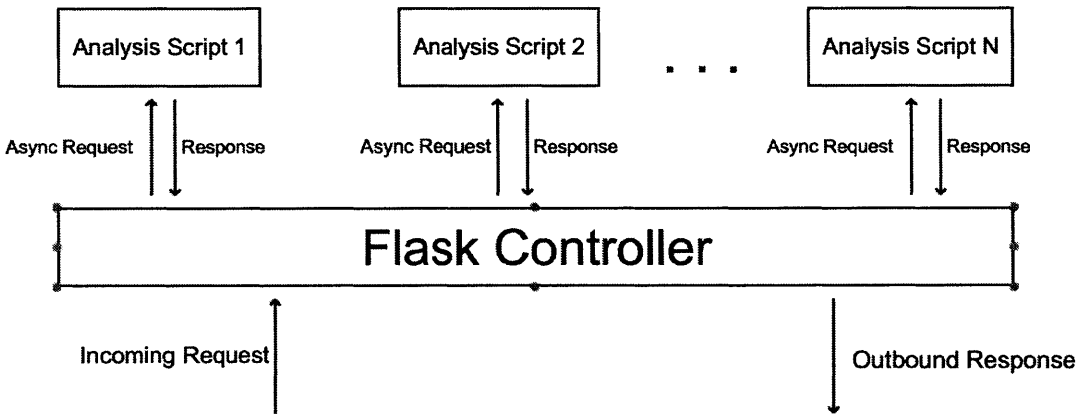

Figure 3-5: Analysis service architecture. The architecture lends itself to multi-processing tasks. Analysis requests are received from the visualization service. The flask controller acquires necessary data from the image service, spatial data service, and molecular data service. After receiving the necessary data, the controller spawns an asynchronous process to run the analysis on the data. Finally, the result of the analysis is output back to the visualization service.

The analysis service allows for the integration of third-party and proprietary ana-

lytical methods to deeply integrate with the rest of the system. The analysis service is optimized to run computationally expensive analytical scripts. By integrating with the platform, these computational methods will have access to image data, spatial data, and the relevant molecular annotations from a single location. When the analysis controller receives a request to run an analytical method, it spawns a new process provided the service has an available CPU core for use. Otherwise, the request is placed in a queue which is dequeued as a core becomes available. Processes created to run analytical methods have full access to the other services offered by the platform.

Object identification is an analysis method that will be used to process new images and mine spatial data from the images. The Visualization Toolkit (VTK) is well-supported software tool that can identify objects and generate mesh representations of the objects that can be saved by the spatial data service. Sequencing read alignment is another analysis method that will be computed using existing aligner software, such as Bowtie, HIVE, or another existing aligner. Data gathered from the aligner will be saved by the molecular data service.

3.5 Spatial Coordinate System Service

The spatial coordinate system is a special analysis service that unifies all of the coordinates from a contiguous experiment into a global coordinate system. The coordinate system is consistent between images and spatial data. Each contiguous experiment can have its own global coordinate space. Distance in global coordinates are related to distance in the physical world. Each unit in any dimension correspond to 10nm in the physical world.

3.6 Spatial Data Service

The spatial data service manages all of the spatial data throughout the system. Spatial data is any data that has a precise location in space with specific coordinates.

Similar to other services, a flask controller is used to monitor requests and send

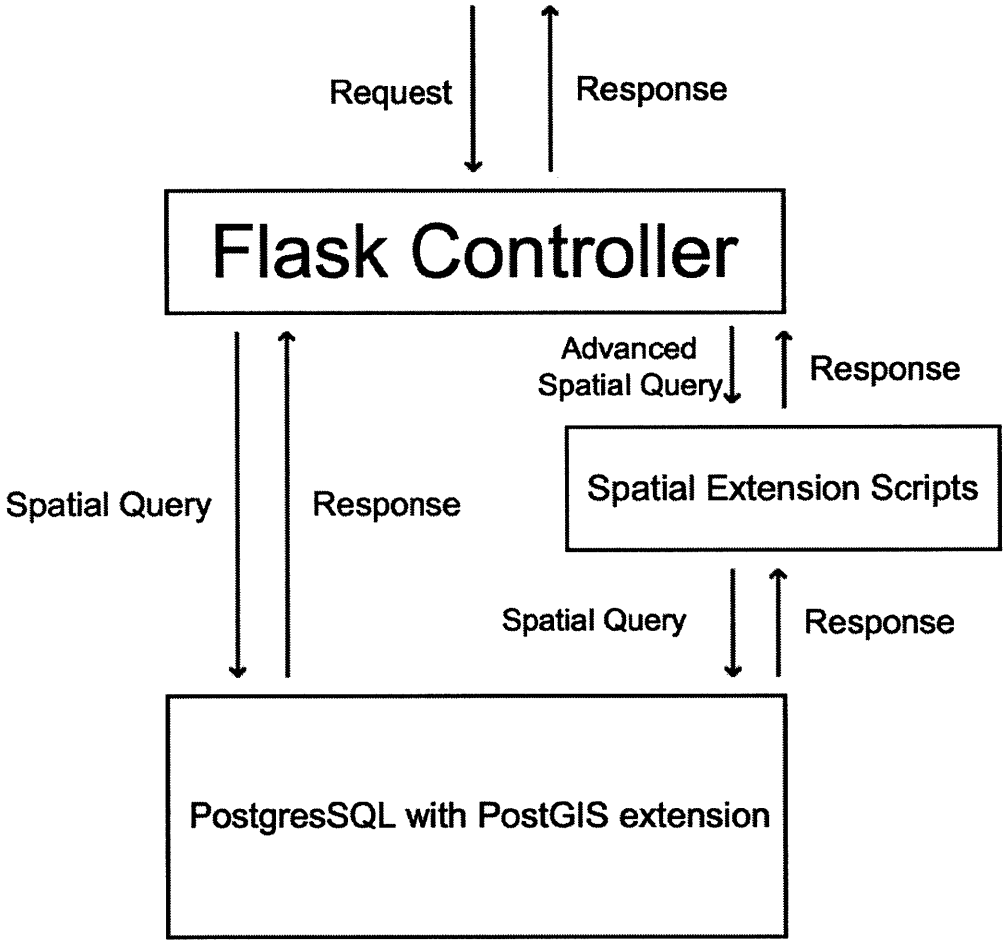

Figure 3-6: Spatial data service architecture. Spatial data is stored in the extended well known binary (EWKB) format in the postgreSQL database. The flask controller organizes requests and responses. The postGIS extension with the SFCGAL engine performs spatial computations

responses. The brunt of the spatial computations is performed by the postGIS extension of the postgresSQL database. Spatial data is stored in an extended well-known binary format (EWKB). The EWKB format extends the well-known binary format (WKB) specified by the Open Geospatial Consortium (OGC) and the SQL / MM standard. The EWKB format optimizes the binary representation of spatial entities. This format enables some common spatial queries, such as calculating a bounding box, to be computed as a series of very fast binary operations. The important difference between WKB and EWKB is that EWKB supports triangulated irregular networks (TINS) and polyhedral surfaces. Both TINS and polyhedral surfaces can be used to represent a mesh in space.

In addition to storing spatial data efficiently, the spatial data service is responsible for processing spatial queries efficiently. The postGIS extension natively supports many of the operations required including: distance operations, intersection checking, and bounding box computations. However, there is some functionality missing. For example, postGIS cannot natively search for objects by containment – to find all objects that exist within another object. To implement the missing functionality, a python wrapper was written around the basic functions offered by the postGIS extension. By composing low level functions together, it becomes possible to extend the possible spatial queries. For example, a containment spatial query was written by taking advantage of the intersection method implemented in postGIS. First, the containment algorithm uses the object's bounding box as a filter to find other objects that may reside inside. Then, for each vertex of each object remaining, 6 rays are extended in the +x, +y, +z, -x, -y, and -z directions. The algorithm counts the number of intersections for each ray and the original object used to define containment. If all of the rays for each vertex has an odd number of intersections with the original object, then the vertex is contained by the object. An object with multiple vertices is contained if all of its vertices are contained. In this way, high level spatial relationship functions can be written. One disadvantage of this method is that the high level spatial relationship functions are not optimized and may take a while to run on large data sets.

3.7 Molecular Data Service

The molecular data service provides a layer of abstraction on top of typical annotations services. Due to the lack of standardization between many annotation services, the molecular data service acts as a single interfaces to many different molecular annotations.

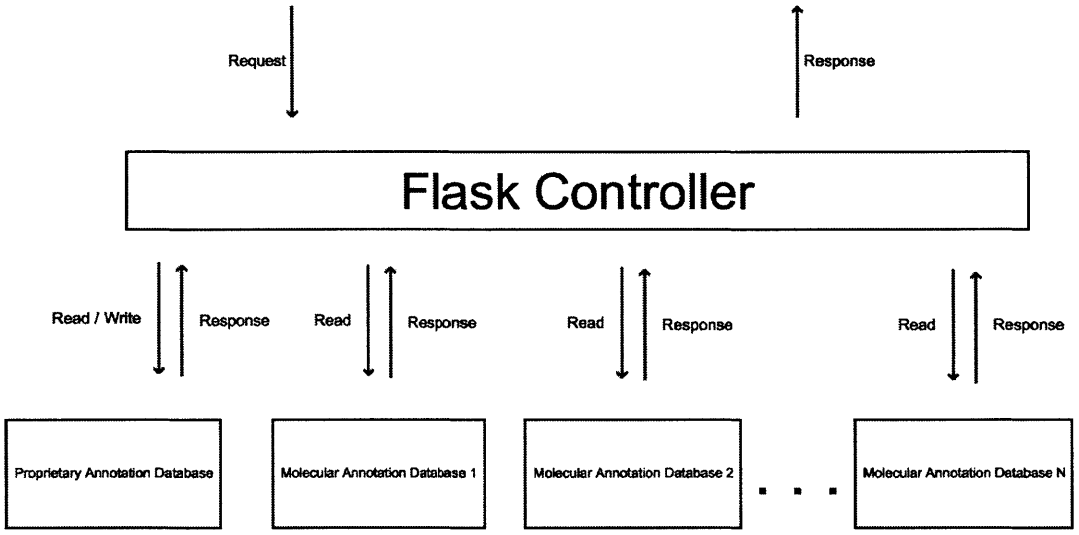

Figure 3-7: Molecular data service architecture. The molecular data service interfaces with external molecular annotation systems, such as the NCBI GenBank

The results from techniques like FISSEQ provide information about where sequences are being expressed and the base pairs that compose the sequence. Sequencing reads by themselves do not have a lot of intrinsic value; however, by aligning the reads against a reference genome and gathering genomic annotations for the aligned sections, it becomes possible to make insights about underlying molecular processes. The molecular data service functions to make it easier to gain insights from sequencing reads. The first component of the molecular data service is the flask controller that defines the API endpoint, processes requests, and returns responses. Requests to the flask controller can communicate with a proprietary postgresSQL database that is used to store sequencing reads, the aligner used, the reference genome used, and the location of the sequencing read in the reference genome. Requests can also retrieve

genomic annotations from a downloaded copy of NCBI's GenBank, DNA DataBank of Japan (DDBJ), or European Molecular Biology Laboratory (EMBL). By providing a layer of abstraction between the annotation services and the rest of the platform, the molecular data service keeps the rest of the platform ignorant of changes that occur in the annotation databases (which can occur daily). Because alignments are stored with respect to their position in a reference genome, it is possible to use a number of annotations services, including ones not list here, to discover the functionality of a specified sequence.

3.8 Visualization Serivce

The visualization service is unique from all other services in that it lives on the client. The visualization service is responsible for providing a graphical user interface (GUI) for a user to easily interact with all of the other services.

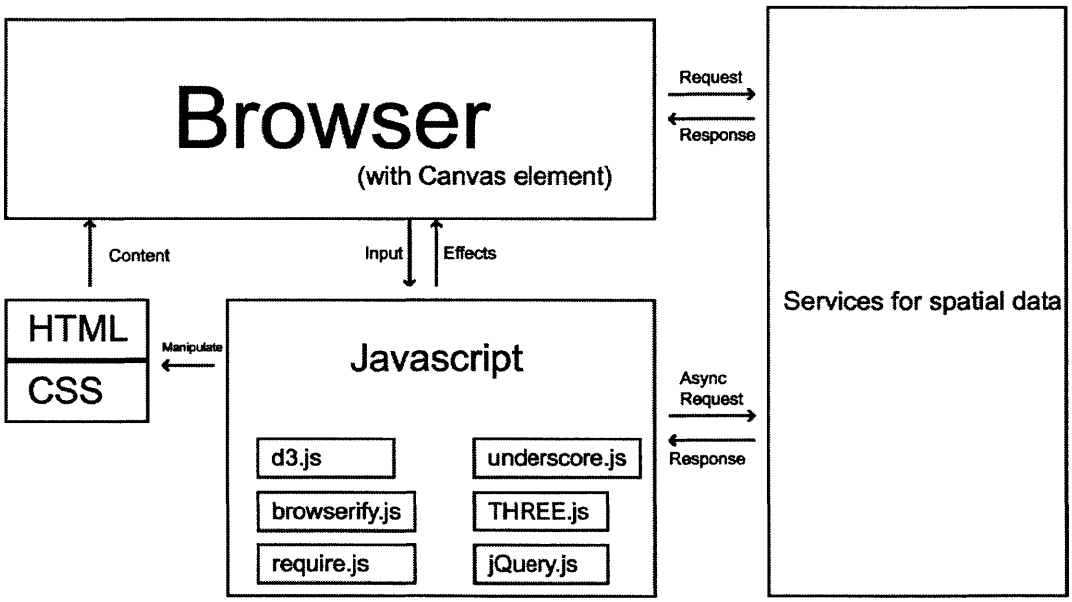

Figure 3-8: Visualization service architecture. The visualization service interacts with other hosted services in the platform. Javascript libraries are used to spawn asynchronous processes. THREE.js is the javascript library primarily responsible for representing the spatial data.

3.8.1 Implementation

The core of the visualization service is written with the THREE.js javascript library. The spatial data is depicted in an HTML5 canvas element. THREE.js is a Javascript wrapper around WebGL, a technology that utilizes the graphics card to quickly compute transformations. Together, THREE.js and WebGL can create the perception of a 3D environment being displayed on a 2D screen. By taking advantage of the graphics card, the visualization service can display computationally expensive scenes with a high framerate.

In order to render a scene, a camera and light must be present. THREE.js is used to instantiate these objects in the scene. The user can switch between using a perspective camera, where distance from objects affects their size, or an orthographic camera, where lengths are not skewed by perspective. Lighting can also be modified to impact how objects appear. There are two main classes of objects to render: an image object and a spatial data object. Using the camera's location in the scene and a specified zoom level, the visualization service queries the image service for relevant images. Relevant images include all images that are needed to fully populate the user's current field of view to a specified depth. Before an image is rasterized, it is piped through the graphics pipeline as a texture. Although most of the graphics pipeline is fixed, two variable parts called the vertex shader and fragment shader are written to highlight important image features while hiding unimportant space. Unimportant space is hidden by decreasing the opacity, or alpha, of the pixel which samples from an unimportant location in the image texture.

The visualization service is also responsible for producing an accurate representation of the spatial data managed by the spatial data service. Spatial data is easier to represent in the scene because it is stored in a mesh friendly format. The simplest object served by the visualization service is a sequencing alignment. These objects are represented as a single vertex in the scene. Additional objects, such as nuclei or membranes, are constructed by drawing their meshes stored in the spatial data service. Each vertex from the vertex map is drawn. Faces are added by walking through

the edge map to create triangles. This flexible format can draw any mesh stored in the spatial data service.

In addition to producing a scene, the visualization service is responsible for allowing the user to interact with the scene. User's can rotate the camera to look at objects from a different angle. User's can also pan around to see objects and images that are nearby. Finally, the user can zoom out to view the macro picture or zoom in to view sequencing alignments. At each zoom level, different features become available to see. User's can click on object to highlight them and identify them. Clicking on the scene shoots a ray from the camera in the direction of the click. The first object that intersect the ray is selected. The visualization service queries the spatial data service and molecular data service for annotations to determine the object's labels. User's can select many objects by defining a region of interest (ROI) selection. A spatial query to the spatial data service searching with GlobalX, GlobalY, and GlobalZ is used to find all objects inside a ROI. Objects displayed and inside the ROI are identified by queries to the spatial and molecular data services.

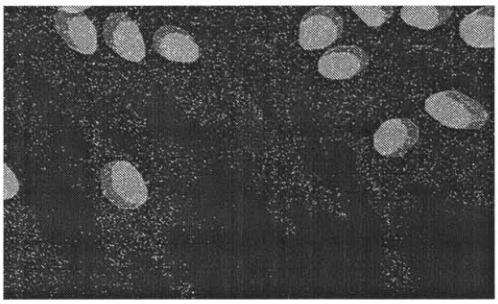

Figure 3-9: Top View

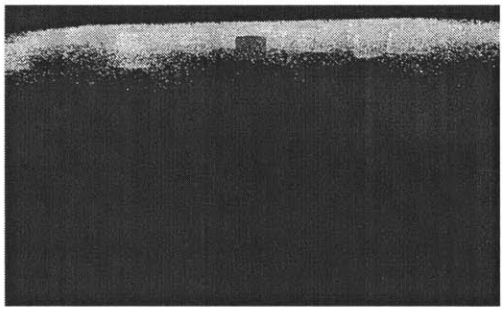

Figure 3-10: Side View

3.8.2 Design Decisions

Building a dynamic 3D scene from a set of images is a fairly difficult task. Without any direct manipulation, only the highest z-layer image (the image nearest the camera) loaded would be visible because the default image opacity is 1. Lowering the opacity of every image would make it possible to see multiple through multiple images, but now objects in the images that should stick out because they are closer would be too

dull and faded. So the issues arises of how to make only select pixels transparent. One way to do this involves loading the stack of images and raycasting through the entire stack. The ray samples images at every point and interpolates the values in between images. After the ray have traversed through the stack, the samples are composed together to form the value of a single pixel on the screen. When the samples are being composed, the weights of the each sample is a factor of how far the ray was when it reached the sample. This weighing scheme will make samples that are far away from the camera have less value and impact on the color of the pixel on the screen. Raycasting is a well documented technique that is fairly effective because each pixel on the screen is only colored once per frame rate. Although raycasting is efficient after the images have been loaded as a stack, it requires a fairly intensive precomputing step of generating an image stack from a set of images. Additionally, as the camera adjusts, the image stack has to grow in memory space and it becomes more expensive to raycast. The alternative is to load images as textures and sample values from the textures to generate pixel values. This does not require an expensive preprocessing step of generating and image stack; however, it can result in a single pixel being rewritten many times in order to generate a scene. The viability of this technique is still being evaluated in scenes with a large dimensions. One benefit of the texture method is that it can be implemented with a gradual loading scheme. The layers closest to the camera load first, which gives the impression that the scene has loaded quickly. Due to the interactivity expected with the visualization service, the texture method was implemented to minimize time on the loading screen.

Chapter 4

Applications

Analyzing spatial molecular data will lead to many new insights in biology. The platform described here will be used by a variety of researchers to make these discoveries possible. This section details potential use cases of this platform.

4.1 Molecular Biologists

Molecular biologists spend a lot of time studying the underlying molecular mechanisms of cellular response to stimuli. Currently, RNA-Seq methods are used to try and elucidate these mechanisms. FISSEQ and other spatial molecular data techniques offer a new alternative to uncover these pathways. One pathway that is currently being researched is the activation of the innate immune system after injury. It is hypothesized that damage associated molecular pattern molecules (DAMPS) are responsible for the activation, but more evidence is needed to accept or reject the hypothesis. Traditional RNA-Seq methods do not have the resolution to easily determine if DAMPS are responsible for the activation of the innate immune system. FISSEQ may offer an alternative with the resolution required to answer the hypothesis. Without the platform for spatial molecular data, researchers will either have to manually analyze images or pipe the data through a series of disparate analytical tools. Either of these alternatives are inferior to the loading the data into the platform for spatial molecular data that unifies all the analytical tools into a usable interface and creates a compre-

hensive digital visualization of the data. Analyzing the data with the platform will be significantly quicker.

4.2 Clinical researchers

In addition to aiding basic science, spatial molecular data can be used to help find new therapeutic drug targets. Recently, its been found that cancers upregulate checkpoint proteins, such as CTLA-4 and PD-1, that inactivate immune cells attempting to attack a cancerous cell. Monoclonal antibodies that target these pathways have been extremely effective at reducing tumor size in patients where PD-1 or CTLA-4 is known to be upregulated and tumor infiltrating lymphocytes (TILs) are present in the tumor mass. Clinical researchers interested in finding new therapeutic targets may use FISSEQ or another method to gather spatial molecular data on the surface of the tumor. Using the developed platform, the spatial molecular data can be uploaded an analyzed with a variety of statistical techniques on the analysis service to try and determine a therapeutic window, such as an abnormally upregulated or downregulated ligand or receptor. The molecular data service can determine the suspected functions of the receptors and ligands if they are known, and the researcher can rank them as potential future targets for drug therapeutics. By providing access to all of these functions with a single interface, the platform for spatial molecular data can help researches spend more time on the important problems.

4.3 Data scientists

Since spatial molecular data is so new, there are not many techniques developed around analyzing this new data that takes advantage of the spatial dimension. Traditional statistical methodologies that are applied to molecular data are still applicable to spatial molecular datasets as well. However, spatial analysis can also be included in statistical analysis to both improve confidence of old metrics and generate new techniques of analysis. Data scientists will be at the forefront of this analysis. For

example, data scientists may develop a statistical technique that can identify regions of interest *de novo*. Typically, researchers look at interfaces between two entities to find areas of potential interest. With spatial molecular data, it would be possible to develop an analytical method that ignores all features except for the spatial distribution of RNA. A novel spatial technique as described would be able to detect regions of interest without bias.

4.4 Developers

Developers may build extra services to interface with the existing platform or improve upon existing services.

Chapter 5

Future Work

There is endless potential for improvements to the platform. One benefit of the service-oriented architecture is that each service can be improved upon individually. Therefore, this section is organized by service again. Most future enhancements will fall into two categories: scalability and features. Scalability refers to improvements that help the service handle enormous loads. Features refer to additional functionality that can be added the the service to make it more useful. Finally, additional services that can extend the system will be discussed as well.

5.1 Raw Data Service

The raw data service is limited by the capacity of the sequencing machine. Scalability is not an important focus for the raw data service in the near future. The feature of highest priority for the raw data service is to provide the sequencing machine with image alignment feedback. Feedback will minimize any mechanical errors that occur as the sequencer processes the input experiments. Additionally feedback can be used to change sampling parameters, repeat imaging of a specific location, or even stop sequencing if the error requires human intervention.

5.2 Image Service

Scalability is an important consideration for the image service because the sequencing machine will generate a lot of high resolution images. Additionally, images retrieval must be quick because the visualization service will query many images to create the visualization. The easiest optimization may be offload the static image hosting to a content delivery network (CDN). CDN will offload the requests to the static images and will serve them faster than the image service. In order to preserve space, the images should be losslessly compressed before uploaded to a CDN. Image editing capabilities would be the first feature to implement on the image service. Loading Imagemagick onto the image service and exposing the API would provide the service image processing capabilities. Useful image editing options include: sharpening, smoothing, filtering, thresholding, deconvolution, object identification, resampling, resizing, cropping, stitching, and windowing.

5.3 Analysis Service

A fully featured analysis service should be capable of running a variety of different analytical programs. In addition to VTK, the analysis service could provide options for object identification including: CellProfiler or NuclearDetector3D. Similarly multiple aligner and reference genomes should be available. Common aligners to include are: SAM, BLAST, BWA, BWA-PSSM, and GEM. Reference genomes GRCh37 and GRCh38 should be available to align against as well. Statistical analysis also be available, such as Fisher's method. Some priority should be placed on developing spatial analysis method because insight from spatial analysis are unique to this data.

5.4 Spatial Data Service

To improve the spatial data service, the high level scripts, such as containment, should be written in the native spatial library CGAL or SFCGAL using C or C++, respectively.

5.5 Molecular Data Service

The molecular data service can continue to be improved by adding more molecular annotation databases to query.

5.6 Visualization Service

There is a lot of room for improvement in the visualization service. One significant improvement would be the ability to add spatial annotation objects to the scenes. Optimizing the scene loading algorithm is crucial to the user experience. Ideally, users will be able to explore the constructed scene freely and should not notice significant loading delays. Unfortunately, the speed of the client's internet is often a limiting factor because building a scene requires pulling in many large image files.

Appendix A

Tables

Table A.1: Suggested TIFF Specification

Tag	Name	Description
256	ImageWidth	Number of pixels per row
257	ImageLength	Number of rows in image
258	BitsPerSample	Number of bits per image
259	Compression	Compression scheme used
262	PhotometricInterpretation	Color space of image
277	SamplesPerPixel	Number of components per pixel
282	Xresolution	Horizontal pixel count per resolution unit
283	YResolution	Vertical pixel count per resolution unit
296	ResolutionUnit	Unit of measurement for X and Y Resolution
306	DateTime	Date and Time image was scanned
315	Artist	Used for ImageProducer
338	ExtraSamples	Description of extra components
269	DocumentName	Document Name
270	ImageDescription	Text string that describes image
42016	ImageUniqueID	unique file identifier

Appendix B

Figures

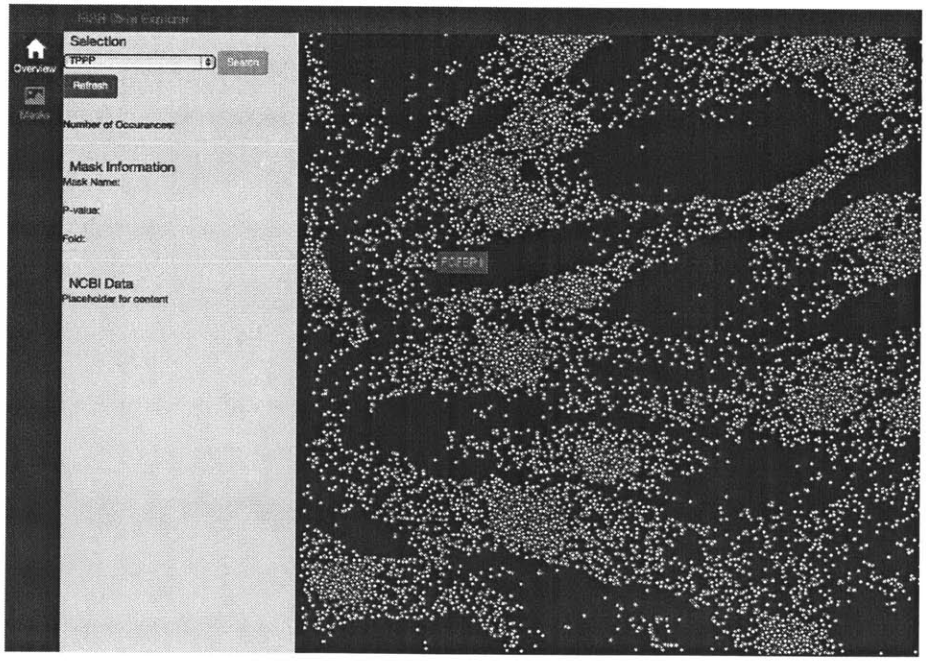

Figure B-1: Visualization service prototype.

Bibliography

[1] Lee JH, Daugharthy ER, Scheiman J, et al. Highly multiplexed subcellular RNA sequencing in situ. *Science.* 2014;343(6177):1360-3.

[2] Michael Bell. Introduction to Service-Oriented Modeling. *Service-Oriented Modeling: Service Analysis, Design, and Architecture.* Wiley Sons.

[3] Alaa M. Riad; Ahmed E. Hassan; Qusay F. Hassan. Investigating Performance of XML Web Services in Real-Time Business Systems. *Journal of Computer Science Systems Biology.* 02 (5): 266?271.

[4] Aji A, Wang F, Vo H, et al. Hadoop-GIS: A High Performance Spatial Data Warehousing System over MapReduce. *Proceedings VLDB Endowment.* 2013;6(11)

[5] Bozorgi M, Lindseth F. GPU-based multi-volume ray casting within VTK for medical applications. *Int J Comput Assist Radiol Surg.* 2015;10(3):293-300.

[6] Chen KH, Boettiger AN, Moffitt JR, Wang S, Zhuang X. RNA imaging. Spatially resolved, highly multiplexed RNA profiling in single cells. *Science.* 2015;348(6233):aaa6090.

[7] PostGIS – Spatial and Geographic objects for PostgreSQL. *PostGIS Project Steering Committee.* http://postgis.net/, 2015

[8] Dirksen J. Learning Three.Js, The JavaScript 3D Library for Webgl. *Packt Pub Limited* 2013.

[9] Dolan DE, Gupta S. PD-1 pathway inhibitors: changing the landscape of cancer immunotherapy. *Cancer Control.* 2014;21(3):231-7.

[10] Kang JW, Kim SJ, Cho HI, Lee SM. DAMPs activating innate immune responses in sepsis. Ageing Res Rev. 2015;