RIAS: an Approach to Provide Internet-Accessible Image Analysis Service

Xinwen Yu¹,², Matthew Laurenson†¹, Takuji Kiura¹ and Seishi Ninomiya*¹

¹) National Agricultural Research Center, National Agriculture and Food Research Organization, Tsukuba, 305-8666, Japan
²) Institute of Forest Resource and Information Technology, Chinese Academy of Forestry, Beijing, 100091, P.R. China

Abstract

An Internet-accessible image analysis service, Remote Image Analysis Service (RIAS), was designed and implemented in this research. The Image Analysis Core, which internally carries out the actual image processing, utilizes a Java code library known as Java Advanced Imaging. Three different inter-process protocols are used for communication between RIAS and client software applications, each serving a different application category. Firstly, Java applications within a host organization’s firewall communicate with the image analysis core using Java RMI, a relatively fast binary protocol. The second category of applications, namely Java clients outside of the organization, access RIAS by sending requests and receiving results through EJB Server over RMI/IIOP. These two protocols are only usable by Java clients. The third category of applications is non-Java client applications, which access RIAS through SOAP-based Imaging Service using HTTP as the transport mechanism. SOAP, as the backbone of Web Service, is a cross platform and a cross language, and hence can be accessed by any languages supporting SOAP. Imaging Service implemented in RIAS supports MTOM, the newest attachment specification. Client applications of Java applet and MS C#. Net were also developed, verifying that RIAS has good performance in three accesses. RIAS is potentially able to be used by mobile devices such as cell phone with camera and PDA with camera. This is important because mobile devices usually lack of ability of floating point computing, by using RIAS, agricultural researchers and farmers may carry out image processing in the field.

Key words
Image analysis, RMI, EJB, Web service, mobile device

Introduction

Imaging is becoming increasingly important in many agricultural fields, and image analysis methods, such as pattern recognition, characteristic measurement, shape evaluation and analysis, are extensively used in both basic and applied agricultural research. Image analysis is computationally intensive and historically was carried out with expensive specialized hardware at dedicated research facilities, and therefore was hardly applied in agriculture practice such as field monitoring. However, in recent years, portable web camera and digital camera have taken the place of the specialized hardware, and become popular in agriculture practice. This helps collect real time images in agriculture field. Particularly, FieldsServer with camera has been applied in many places for field monitoring, indicating that online image processing and analysis is possible. The inevitable increasing of this kind of field monitoring scheme demands a lot of real time image processing and analysis. Currently, image processing and analysis is still very hard for common users to make use of; various effective algorithms and programs are stored on the hard disks of many image analysts and researchers without being applied. If there is an approach to make those buried image analysis methods shareable among common users, it will be of great importance to real time field monitoring. To our knowledge, there are some discussion rooms on the web (Palaniappan 1998, Franz and Goller 1998), but most focus on management of distributed image data (Takahiro 1998, University of Alberta 2000, Rosen-garten 2002), and are geo-science related.

The objective of the work reported in this paper was to develop an image analysis service for image software applications. The
service provides a mechanism for agricultural and biological image analysts to contribute their expertise and makes their verified image algorithms sharable; hence the developers for image software applications can easily make use of the algorithms provided by the service and concentrate on the software designation. The intention was to provide application developers with the functionality of an image analysis code library by carrying out the actual image processing on a separate server from the device.

Pre-consideration

For the development of the service there are several requirements. The basic requirement is to provide a mechanism of easily adding wide range of image analysis functions for image analyst. In the core of image analysis code library, all the image analysis functions are encapsulated into objects. As long as an image analyst developed a new image analysis function which is encapsulated into objects, it can be added into our code libraries. Therefore, the service should have a clear, well-documented application programmer’s interface (API) so that developers of image analysis algorithms can write their own image analysis object, and developers of image software applications can easily incorporate our service access into their applications. Secondly, the service should be accessible from a wide range of programming languages so that programmers do not have to learn a new programming language in order to use the service. Thirdly, the service needs to be able to support multiple simultaneous users, and image data need to be transferred to and from the service over limited-bandwidth Internet links. Therefore, the service needs efficient mechanisms for handling such data.

There are a variety of protocols for handling this kind of inter-process communication that our service will require and we choose Java Remote Method Invocation (RMI), a binary protocol, as the base communication mechanism. However, RMI uses port 1099 is often blocked by organizational firewalls. This can be solved by adoption of wrapper servlet or EJB technology. Furthermore, Java RMI, servlet and EJB services can only be accessed by Java clients. An XML-based protocol called Simple Object Access Protocol (SOAP) (Box et al. 2000) has begun to be widely used and became the backbone of Web service (Apache 2003). SOAP provides similar functionality to Java RMI and is independent of any particular programming language. Many programming languages, e.g. VB, C#, Delphi and even Excel, have toolkits allowing them to communicate using SOAP and interoperability between toolkits is rapidly improving. So an adoption of SOAP based Web service can provide interface to different programming languages for our image services. Fig. 1 shows two examples of toolkits translating applications’ native language calls into SOAP method invocations across the network to a Web service, and converting the returned SOAP objects back into native objects.

As Fig. 1 indicates, most SOAP implementations pass XML messages using the standard Web protocol (HTTP). HTTP is by default allowed to pass through organizational firewalls. The widespread development of SOAP toolkits, and SOAP’s compatibility with firewalls together promises to user in a new era of XML-based data exchange, just as HTML enabled the growth of the original World Wide Web (Newcomer 2002).

System design and implementation

According to the pre-consideration, a system of Remote Image Analysis Service (RIAS) was designed and implemented to allow users to develop image analysis applications by accessing RIAS using different programming languages, and also to provide an approach for sharing of image analysis methods.
RIAS: an Approach to Provide Internet-Accessible Image Analysis Service

Design

RIAS consists of ImageAnalyzer, Imaging Service and client applications. ImageAnalyzer, the image analysis core of RIAS, is composed of image code library, RMI Server and EJB server. Imaging Service is a web service exposing the function of ImageAnalyzer to other programming languages. The client applications send processing instructions and images to RIAS and receive processed images.

As introduced in the pre-consideration, three different inter-processes are used for communication between RIAS and client software applications, each serving a different application category. Much of RIAS’ structure derives from the need to provide these three communications paths.

First and second categories are both Java applications. Firstly, Java applications within a host organization’s firewall communicate with ImageAnalyzer by accessing RMI server, a relatively fast binary protocol.

Secondly, Java applications running outside the host organization typically communicate with RIAS through two firewalls—that of the organization hosting RIAS, and that of the organization hosting the client application. Java RMI uses a range of dynamically-assigned Internet ports, and it is not practical for each potential user to have to persuade their firewall administrator to open their firewall to extent RMI requirement. In this case, the second category of applications, namely Java clients outside of the organization, communicates with RIAS by accessing EJB server over RMI/IIOP. This approach avoids each potential user having to persuade their firewall administrator to open the necessary ports because most organizational firewalls are set up to freely pass IIOP requests.

The third category of applications is non-Java client applications, which access the system through a SOAP-based Web service. SOAP Web service uses HTTP as the transport mechanism and will not be blocked by organizational firewall.

Although Java applications could also potentially use the SOAP web service, accessing to EJB or RMI server is encouraged, because conversions to and from SOAP format are relatively expensive in terms of their processing requirements, and the SOAP format is also relatively verbose and bandwidth-consuming. Providing higher-performance binary access options for Java applications carries little cost because RMI access is a side benefit of separating the Web service from the image analysis core.

Internally, ImageAnalyzer utilizes a Java code library in which various image processing and analysis algorithms are accordingly defined in a simple style, so that potential image analysts could incorporate their algorithms into this library.

Implementation of ImageAnalyzer

ImageAnalyzer consists of image code library, RMI server and EJB server. Image code library, the core function module of ImageAnalyzer, is composed of image operators developed based on Java Advanced Imaging (JAI, Sun Microsystems 1999). JAI is an extension package of Java 2 Standard Edition (J2SE), and is distributed separately to J2SE. Users must install a copy of JAI in their computing environment to make use of the functionality of JAI. According to our experience, many users are reluctant to have an installation of JAI. Furthermore, JAI is to some extent complicated, as it introduces many new concepts and image classes (e.g. TiledImage, PanoramaImage, ImageOp) that are difficult to understand and utilize in a short time.

We therefore developed a set of wrapper classes using BufferedImage as an image object to simplify the usage of JAI, since BufferedImage is the main image class in J2SE. By using these wrapper classes, users without any knowledge of JAI can make use of the powerful imaging functionalities of JAI. We also added some useful operators we developed ourselves with the same form of these wrapper classes.

The RMI server is actually a control program responsible for analyzing the request from client applications and dispatching request to corresponding image operator. Image operator returns processing result to the control program and then the control program returns the result to the client application.

An image object will be transferred between client and server application, and Java demands object serialization when trans-
BufferedImage is not serialized; we also developed 3 serialized versions of BufferedImage under the interface of SerializableImage, namely, PackedImage packing BufferedImage to byte array, UnPackedImage decomposing BufferedImage to serialized objects, and URLImage, representing images distributed on the Internet. Image file formats, such as JPEG, PNG, GIF, BMP, TIF, etc, are supported in this system.

Fig. 3 demonstrates the mechanisms of image serialization.

The EJB server is implemented using recently released EJB3 technology, which is much simpler than older version (Raghu 2005). By using JBOSS EJB3 (JBOSS 2006), an implementation of the latest specification in J2EE framework (Sun Microsystems 2005), the implementation of EJB3 server based on RMI is very easy, just exposing the RMI remote interfaces to EJB3 style and implement these interfaces.

Imaging Service

There are several choices of Web service implementations, for example, JAX-WS 2.0 (Sun Microsystems 2006), JWSDP1.6, Apache AXIS, etc. We choose JAX-WS 2.0 as the toolkit to implement Imaging Service, because JAX-WS 2.0 supports the newest specifications related to Web Service. The most important reason to adopt JAX-WS 2.0 is because it, via JAXB, adds support for MTOM (W3C 2005), the new attachment specification. The Client of Imaging Service usually needs to send an image with processing request to and receive the processed image back from Imaging Service, an efficient image transferring mechanism is very important to the performance of Imaging Services. Before the release of MTOM, most Web Service vendors adopted specification of SOAP with attachment which is based on MIME model, and this model proved to be the hardest part of Web service (Loughran 2003). Additionally, Microsoft Web Service defined its own attachment specification based on DIME (Microsoft 2001) which is different from MIME (Gailey 2002). In this situation, it is very hard to realize the interoperability between different programming languages. But it appears that everyone supports MTOM, and so attachment interoperability should become a reality.

JAX-WS 2.0 simplified the development of Web Service by adopting many new handling models, especially by heavily relying on Java Annotation features. The implementation of Imaging Service using JAX-WS 2.0 included 3 steps.

The first is to define an implementation class that communicates with RMI server to transit request and result between the server and client. This class must have an annotation of @WebService before the class name, so that JAX-WS 2.0 tools can generate required artifacts. Following code demonstrates the class definition of Imaging Service.

```java
@WebService(endpointInterface="net.agmodel.imageserver.ws.jaxws.ImageServiceEndpoint")
public class ImageSoapEndpointImpl implements ImageServiceEndpoint{
...
}
```

The second is to generate portable artifacts for web service execution. Actually, this step is optional. JAX-WS’s deploy tool automatically generates these artifacts if they are not bundled with a deployable service unit during deployment of the web service. We can manually generate these artifacts using WSGEN tool. The generated artifacts include a WSDL (Web Service Description Language) along with a schema file, which defines the schema for WSDL file.

The third is to package and to deploy the Imaging Service which is ready to use. Imaging Service is bundled as a servlet and packaged in Web Archive (WAR) file, and then deployed to Tomcat web server.

Client side JavaBeans

We also developed several small Java components called JavaBean, which are very useful for connection to ImageAnalyzer (termed the ImageRequestBean in our project) and parameter collection (ImageBeans) necessary for image processing. The client application uses ImageRequestBean to make request (wrapped with ImageServerRequest) for image processing methods when receiving any messages from ImageBeans, and then sends the request to ImageAnalyzer. ImageRequestBean, extended from ServerRequestBean which is a class of MetBroker (Laurenson et al. 2001), communicates with the ImageAnalyzer core, preferably using RMI. When access to RMI is blocked by a firewall, it will try to connect to the EJB server.
RIAS: an Approach to Provide Internet-Accessible Image Analysis Service

**RIAS Capabilities**

RIAS supports all of the file formats supported by JAI, such as JPEG, PNG, GIF, BMP, and TIF. At present, ImageAnalyzer provides more than 16 types of operators (for detailed information refers to http://www.agmodel.net/ImageAnalyzer) and additional operators are under development.

**Client Applications**

A Java-based image analysis applet was developed in order to verify the performance of Image Analysis Core. This applet can be evaluated at http://www.agmodel.net/ImageAnalyzer. The applet communicated directly with RIAS using the RMI interface if possible or the EJB Server if RMI is blocked by firewalls. To facilitate experimentation by users, RIAS was extended to make a set of sample images available to such client applications.

The applet incorporates a set of reusable user-interface components including an image display canvas and a tabbed control panel that displays the list of available operations and lets the users set their parameters through controls such as sliders and combo boxes. To allow users in different countries to evaluate RIAS, the button and tab labels in the applet display in each user’s preferred language as determined by Java. Currently text in English, Chinese and Japanese is maintained and accessed using the ResourceServer system (Laurenson et al. 2002).

Fig. 4 shows screen shots of the applet. In Fig. 4A the applet displays a sample image selected from those available from RIAS. The control panel in the lower part of the screen is composed of 11 tabbed panes; each pane controls the behavior of one type of image operator. Users can apply an operator by selecting the relevant tab and manipulating the controls on that tab. In general, the image is submitted to RIAS for processing each time the user makes a selection from the controls.

Fig. 4 shows screen shots of the applet accessing RIAS through the Java RMI protocol.
Fig. 5 An application of Microsoft .Net accessing RIAS through the SOAP protocol

3.0, Fussell 2005). Fig. 5 shows several screen shots of this application. When user selects an image processing command from menu (for example, Rotate, as showed in Fig. 5A), the application queries the user for parameters relevant to this operation (Fig. 5B), creates an RIAS request with the raw image and processing instructions, and then displays the returned processed image (Fig. 5C)

Documentation

Java provides a very simple and efficient way to produce and maintain documentations along the development. This guaranteed that RIAS has a well organized documentation. Documentation for RIAS is available from the web site http://www.agmodel.org/ImageAnalyzer to help potential client application developers to utilize RIAS. The documentation explains the architecture and inner structure of ImageAnalyzer, including instructions for client application development. All the Java classes are documented in JavaDoc format. The Web service is described in a standard way by a Web Services Definition Language (WSDL) file at http://www.agmodel.org:8080/ImageAnalyzerws/ImageService?wsdl.

Although it is not necessary for client application developers to understand the inner workings of RIAS, the code of ImageAnalyzer may be published as an open source project in order to further encourage uptake. We recognize that application developers are unlikely to utilize RIAS for “mission critical” applications unless they have some control (either physical or contractual) over the operation system.

Discussion

We developed the RIAS system to provide a pool for agriculture image algorithms and also to provide consistent access to these algorithms via Internet, so that agriculture image analysts can embedded their imaging algorithms to the code library of RIAS, and developers of application software can make use of these algorithms instead of developing imaging algorithms themselves. Adoption of Web Service technology enables applications of RIAS developed from different programming languages (Java, MS C#. Net) access RIAS efficiently.

The RMI and EJB server’s remote interfaces do not include any JAI classes, enabling Java client and Imaging Service have the functionality of JAI and imaging algorithms embedded by image analysts even without the installation of JAI package.

J2ME Web Service APIs (J2ME WSA) have become available and enable J2ME devices to be web services clients, providing a programming model that is consistent with the standard web services platform (Sun Microsystems 2004). This means that mobile devices with camera can be the clients of RIAS. However, the overall computational power of these mobile devices is constrained by battery life/energy consumption issues, limited memory, and sometimes a lack of floating point arithmetic (Gusev 2005). These devices are not yet suitable platforms for stand-alone image analysis, but they can be easily taken into the field by farmers and researchers to capture digital images and communicate with other computers. There would be considerable benefit in terms of timeliness and convenience if image analysis could be routinely used within software applications being run on mobile devices in the field. RIAS provides an approach for these mobile devices to have image analysis capability.

Cell phone with digital camera is very popular in Japan, and is becoming popular in the world. Some cell phone and PDA based agricultural information systems are available (Sugawara 2001, Otuka et al. 2003, Otuka & Yamanaka 2003, Sasaki et al. 2003), but these systems are all text based. Using RIAS, we can develop image based decision support system. Farmers acquire images in the field and send images to RIAS; RIAS can analyze the image and make decision on what happened in the field and what to do for the situation. This is very suitable for pest diagnosis, recording of pesticide utilization, crop development analysis and so on.
Since the images are acquired in the field, and farmers can get decision immediately after sending images to RIAS, there is no time lag for the decision making.

Both agriculture and image analysis use their own sets of terms, and these terms sometimes differ from area to area, researchers to researchers. Current version of RIAS can only deal with terms for agriculture and image analysis which are adopted by our organization. An adoption of ontology function will be a good solution for the interpretation of different terms; we are developing the ontology and will integrate it into RIAS in the near future.

Acknowledgements

This research was supported by the Japan Society for the Promotion of Science (JSPS) and Special Coordination Funds for Promoting Science and Technology.

References


Received June 25, 2007
Accepted August 29, 2007

Environmental information