FINDING IMAGES OF TERRAIN FEATURES IN LARGE DATABASES COLLECTED BY ROVERS

Debra Schreckenghost [1], Tod Milam [2], David Lees [3], and Terrence Fong [4]

[1] TRACLabs, 16969 N Texas Ave, Suite 300, Webster, TX 77598, ghost@ieee.org
[2] TRACLabs, 16969 N Texas Ave, Suite 300, Webster, TX 77598, tmilam@traclabs.com
[3] Carnegie-Mellon University Silicon Valley, Moffett Field, California 94035 david.s.lees@nasa.gov
[4] NASA Ames Research Center, Bldg 269, Moffett Field, California 94035 terry.fong@nasa.gov

ABSTRACT

NASA's exploration and scientific missions will produce terabytes of information. A key challenge for these missions is improving utilization of this information. For example, geologists are often interested in finding images of terrain features in an area. The DELVE (Database Exploration to Locate Visual Elements) software searches an image database to find the images most likely to show a terrain feature. It does this by determining whether the feature location is within the expected viewing area of the camera when the image was taken. In this paper we describe the DELVE software and the evaluation of DELVE using images collected at a NASA field test in 2010.

1. INTRODUCTION

NASA's exploration and scientific missions will produce terabytes of information. As NASA enters a new phase of space exploration, managing large amounts of scientific and operational data will become even more challenging. Robots conducting planetary exploration will produce data for selection and preparation of exploration sites. Robots and space probes will collect scientific data to improve understanding of the solar system. Satellites in low Earth orbit will collect data for monitoring changes in the Earth's atmosphere and surface environment. Key challenges for all these missions are understanding and summarizing what data have been collected and using this knowledge to improve data access.

Images taken by a planetary rover typically are indexed by the location and heading of the rover-mounted camera when the image was taken. Geologists, however, are often interested in images of terrain features. The DELVE (Database Exploration to Locate Visual Elements) software identifies the images most likely to show a terrain feature by determining whether the feature location is within the expected viewing area of the camera when the image was taken (called a *viewing cone*).

DELVE searches large image databases to find images likely to show a terrain feature identified by the user (called a *feature query*). The user marks a feature of interest on a satellite map. The feature query algorithm identifies the images most likely to show this feature by determining whether the feature location is within a viewing cone with its apex at the rover location when the image was taken. The feature query algorithm searches image metadata generated from rover telemetry while the images are collected. This metadata describes the rover and the environmental conditions when the image was taken, and includes pre-computed viewing cones for each image in the database. This approach provides a rich set of image metadata that can be searched quickly.

The DELVE software has been tested with images collected by NASA's K-10 rover during simulated remote robotic operations at the Haughton Mars Project (HMP) in the summer of 2010. The Intelligent Robotics Group (IRG) at NASA Ames Research Center developed the K10 rover. During the HMP field test, K10 executed plans built by a science team that specified locations where images should be taken. Images collected by the rover were transmitted to a remote operations site and stored for use by the science team. The K10 rover collected images from three different instruments during this robotic field test: a panoramic imager (PanCam), a microscopic imager (MI), and a Lidar. The PanCam is a 12-megapixel digital Canon PowerShot G9 camera mounted on a pan-tilt unit. The PanCam is operated at 350 rad/pixel, comparable to the Mars Exploration Rover (MER) Pancam (280 rad/pixel). The MI also uses a Canon PowerShot G9 camera attached to K10 with a fixed downward-facing mount. At the highest resolution, the MI provides 33 microns/pixel at the ground, comparable to the spatial resolution of the MER Microscopic Imager. The Lidar is an Optech Intelligent Laser Ranging and Imaging System (ILRIS-3D). While Lidar images are not accessible in the HMP image database, digital single shot images taken by the PanCam in the same pointing direction as the Lidar are available.

In the remainder of this paper we describe our feature

query approach to search databases for images of terrain features, the evaluation of this approach for images collected at a NASA field test in 2010, and our conclusions and future work.

2. APPROACH

Images taken by the rover are indexed by the location and heading of the rover when the image was taken. Thus retrieving images by this location will not necessarily show a feature near this location. The feature query algorithm identifies the images most likely to show a feature location by determining whether the feature location (defined as a *feature line* and *viewing direction*) is within a viewing cone with an apex at the rover location. Specifically a match is found when (1) either end point of the feature line is contained in the viewing cone or (2) when the feature line crosses a boundary of the viewing cone. The size and shape of the cone depends upon the type and resolution of the image. The algorithm also considers alignment between the rover heading when the image was taken (i.e., the pointing direction of the camera) and the desired viewing angle. An image is considered likely to show a feature when the difference between these angles (θ) is less than 45 degrees. We describe our approach for implementing the feature query software in this section

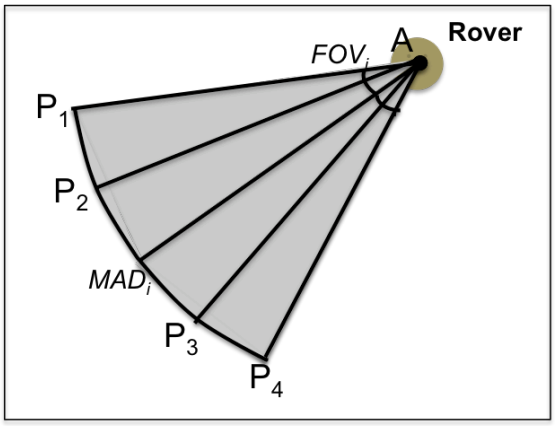
2.1. Image Metadata

The DELVE software automatically derives XML metadata about images using rover telemetry streamed as images are collected. The software we developed for rover performance monitoring [3, 4] was modified to automatically derive image metadata. This metadata describes the rover and the environmental conditions when images were taken. It includes pre-computed viewing cones for each image in the database. This approach provides a rich set of metadata characterizing images that can be searched quickly. Metadata derived for each image includes the following:

- Rover plan and task to take the image,
- Location of the rover when image was taken,
- Date and time when the image was taken,
- Whether the data collection task ended normally, and
- Camera settings (resolution, field of view) when the image was taken.

A viewing cone defines the region expected to be visible by the camera at the specified settings. A viewing cone is computed for the rover location and camera settings used to take each image. It is modelled as a multi-point polygon in the XML metadata. The polygon is defined as a segment with a radius called the *Maximum Acuity Distance (MAD)* and an angle called the *Field of View (FOV)* about the rover location when the image was taken. The number of points in the polygon depends upon the image FOV: (1) Lidar – 5 points, (2) PanCam Narrow – 5 points, (3) PanCam Medium – 7 points, and (4) PanCam Wide – 11 points. Figure 1 shows the FOV and MAD used to compute the viewing cone for each image profile.

Image Profile	FOV	MAD
Lidar Scan Low Res	40 deg	80 m
Lidar Scan Med Res	40 deg	100 m
Lidar Scan Hi Res	40 deg	150 m
Pancam Wide Angle	180 deg	30 m
Pancam Med Angle	120 deg	50 m
Pancam Narrow Angle	40 deg	80 m
MicroImage	360 deg	1 m



Information pre-computed about image profile
- FOV_i = field of view for the image type i
- MAD_i = maximum acuity distance for image type i

Figure 1. Viewing Cone Parameters

2.2. Feature Query Specification

To find images showing a terrain feature, the user first locates the area of interest on a satellite map in Google Earth. He or she then defines the extent of the feature by drawing a line over it (marked by a fuchsia line in Figure 2), and choosing which side of the line to view (marked by a yellow arrow in Figure 2).

This feature line and viewing direction are converted to parameters that are passed to the search algorithm. The center point *fc* of the feature line (*f1, f2*) is computed, where points are defined in latitude and longitude. The width of the feature line (*f1, f2*) is computed in meters. The viewing angle φ {-180 to 180 degrees} is computed as the angle of rotation between North and a line perpendicular to the feature line.

Two additional parameters are passed to the search algorithm: the maximum allowable distance d_{max} between the feature (default setting of 250 meters) and the rover, and the maximum obliqueness angle θ_{max} (default setting of 45 degrees). The obliqueness angle θ is the angle between the rover heading (i.e., the camera viewing direction) and a perpendicular to the feature line. The angle of obliqueness θ is computed as φ - δ. Figure 2 illustrates the geometry of the search parameters.

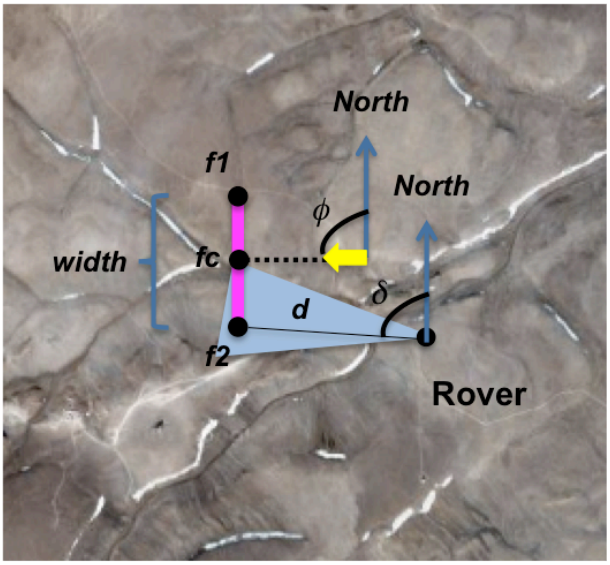


Figure 2. Geometry of Feature Query Parameters

2.3. Feature Query Search Algorithm

The feature query server software implements the search algorithm. It receives the following search parameters when making a query:

- *fc*: latitude and longitude of the centerpoint of the feature line,
- ϕ: viewing angle of the feature,
- *width*: length of feature line,
- d_{max}: maximum distance between feature and rover, and
- θ_{max}: maximum angle between the rover heading and a perpendicular to the feature line.

The search algorithm identifies the images likely to show a terrain feature by performing a series of tests on the image metadata. The metadata for an image must satisfy all tests to be selected as likely to show the feature. The tests are performed in the following sequence, where each subsequent test is only performed on images that pass all prior tests:

1. ***Distance Check***: find images where the shortest distance between the rover and the feature line is within d_{max}
2. ***Coverage Check***: find images where the feature line intersects or is contained in the viewing cone of the image
3. ***Obliqueness Check***: find images where the angle between the rover heading and a perpendicular to the feature line is within θ_{max}

Distance Check: The first test finds all images where the shortest distance *d* between the feature line (*F1, F2*) and the rover location *A* is less than or equal to d_{max}. The shortest distance *d* is computed as shown below. Images where $d_i <= d_{max}$ are passed to the Coverage Check. Figure 3 illustrates the geometry of this test.

$$x_d = x_{f1} + u\left(x_{f2} - x_{f1}\right) \quad (1)$$

$$y_d = y_{f1} + u\left(y_{f2} - y_{f1}\right) \quad (2)$$

$$u = \left[\left(x_A - x_{f1}\right)\left(x_{f2} - x_{f1}\right) + \left(y_A - y_{f1}\right)\left(y_{f2} - y_{f1}\right)\right] / \left[\left|F1 - F2\right|^2\right] \quad (3)$$

$$d = \left[\left|A - Fd\right|^2\right] \quad (4)$$

where

$Fd = minimum\,distance\,point\,on\,feature\,line = \left(x_d, y_d\right)$

$d = distance\,from\,rover\,A\,to\,Fd$

$A = rover\,location\,when\,image\,taken = \left(x_A, y_A\right)$

$Fi = endpoints\,of\,feature\,line, i = \{1, 2\}$

$F1 = \left(x_{f1}, y_{f1}\right)$

$F2 = \left(x_{f2}, y_{f2}\right)$

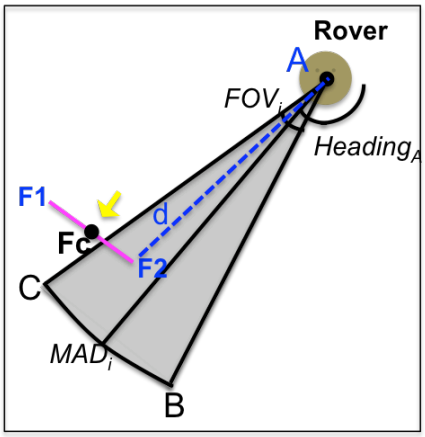


Figure 3. Geometry of Distance and Coverage Checks

Coverage Check: For the second test, a viewing cone is retrieved for each image taken within the minimum

distance d_{max}. Java polygon methods are used to determine whether the feature line either intersects the viewing cone or is contained within the viewing cone. Figure 3 illustrates the geometry of coverage checking when the feature line intersects the viewing cone.

Obliqueness Check: For the third test, an obliqueness check is performed for all images where the feature line either intersects its viewing cone or is contained within its viewing cone. The obliqueness check determines whether the viewing angle of the image is no more than 45 degrees from a perpendicular to the feature line. The obliqueness check consists of the following steps:

1. Compute the shortest distance d between the rover location A and the feature line. This corresponds to the shortest distance in $\{d_{ci}, d_p, d_{fi}\}$. In Figure 4 this is d_p, the length of the line between A and p.

   $$d = \min\{d_p, d_{ci}, d_{fi}\} \qquad (5)$$

   where

   $d_p = |p - A|^2$ = *distance from A to intersection of perpendicular p with feature line*

   $d_{ci} = |ci - A|^2$ = *distance from A to intersection of viewing cone ci with feature line*

   $d_{fi} = |fi - A|^2$ = *distance from A to endpoints fi of feature line contained in viewing cone*

2. Compute the heading δ from North of the shortest line between A and the feature line determined in step 1. δ corresponds to viewing angle of the image. The Geotoolkit (geotoolkit.org/index.html) open source library was used to implement this step.

3. The desired viewing angle ϕ was passed in as a query parameter.

4. Compute the angle of obliqueness θ as the difference between the viewing angle of the image δ and the desired viewing angle ϕ.
   $\theta = \phi - \delta$

5. If $\theta <= \theta_{max}$ (45 deg), the image is likely to show the desired feature and is returned as a query result.

Figure 4 shows the geometry of the obliqueness test.

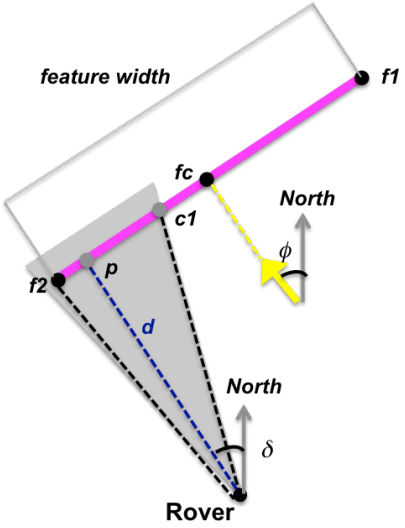


Figure 4. Geometry of Obliqueness Check

2.4. Feature Query Results

The feature query server returns the results of a search as a XML data structure identifying all images likely to show the desired terrain feature. These results specify (1) the search parameters passed to the feature query server, (2) a list of image IDs showing the feature, (3) metadata for each image ID describing conditions when the image was taken (image type, rover location, rover heading, the distance between rover and feature, and the time the image was taken), and (4) the points that define the viewing cone for each image in the list. The feature query server optionally returns these search results formatted in GeoJSON.

The feature query results can be displayed in two ways – in the IRG's xGDS Share web display or in a new web display developed for DELVE. We describe these displays below.

xGDS Share Display: The Intelligent Robotics Group (IRG) developed the Exploration Ground Data System (xGDS) Share software. The Share software provides a web interface to the database of images collected by the K10 rover. New software was added to xGDS for specifying feature query parameters from Google Earth within the xGDS software and passing them to the feature query server. The user identifies a feature in Google Earth by drawing a line over it (indicated by a fuchsia line in Figure 6), then choosing which side of the line to view (indicated by a yellow arrow in Figure 6). The locations of images returned by the feature query are displayed as icons overlaid on a satellite image in Google Earth and as thumbnail images in a table. The GeoJSON query results are passed to the Share function that retrieves the appropriate KML icons

for each image. The image type is used to determine the type of icon and the heading used to rotate this icon for Pancam and Lidar images. These KML icons are overlaid on the satellite image in the Google Earth plug-in. Thumbnails of these images are retrieved and shown in a table to the right of the Google Earth satellite image. Clicking on a thumbnail shows the image at a higher resolution. Figure 6 shows an example of the xGDS web display modified for the feature query.

DELVE Web Display: A new web display was developed to view feature query results. This display shows the locations of images satisfying the query as viewing cones overlaid on a satellite image in Google Earth. Such a display enables the user to see how the image viewing cones overlap with the selected feature. Similar to the xGDS Share web display, the user marks a line over the feature of interest and then selects which side of the feature to view. The resulting query specification is shown as two dark blue lines perpendicular to each other (see Figure 7 below). The long blue line identifies the feature and the short blue line identifies the viewing direction. Results returned from the feature query server are shown as viewing cones, using KML computed for each image by the feature query software. The images returned by the feature query also are shown as thumbnails in a table to the right of the satellite image map. Next to each thumbnail image, metadata describe the conditions when the image was taken. The images in the table can be sorted using any of this metadata by clicking on the column header for the metadata. A checkbox in the left-most column is used to selectively show/hide the view cone for an image, useful in de-cluttering the map to see a specific viewing cone. The table of thumbnails can be used to preview and compare images in the query results. The DELVE web display enables the user to see how the image viewing cones overlap with the selected feature. The ability to visualize the spatial relationship between the viewing cone and the feature helps the user understand why an image satisfied a query and which of the returned images are most likely to have the best view of the feature. Figure 7 shows an example of the DELVE web display for images collected at HMP on August 6, 2010.

3. EVALUATION

The DELVE software was evaluated using a database of images collected by the NASA Ames Research Center's K10 rover at the Haughton Mars Project (HMP) field test at Haughton Crater, Devon Island, Canada, in 2010 [1]. The images at HMP 2010 were selected as representative of database of images collected by the planetary rover. HMP data are stored in the IRG's exploration Ground Data System (xGDS) Share database. For this evaluation the DELVE software was integrated with the xGDS Share software.

Rover telemetry recorded at HMP 2010 was used to build XML files of metadata for the images in the Share database. This metadata included the KML describing the viewing cone for each image in the database. The DELVE software searches these XML metadata files to find images that match the feature query input arguments.

The feature query functionality is accessed through a feature query server implemented in Java. It is possible to make a query in two ways: (1) append the query arguments to the end of the server URL, or (2) pass the arguments as an XML structure that is posted to the server URL. Both approaches specify the following arguments: (1) latitude and longitude of the feature mid-point, (2) width of the feature line, (3) desired viewing angle in degrees, (4) maximum distance between the rover and the feature in meters, and (5) format of the query results (JSON, XML, or both). These parameters are passed to a javascript function that constructs an http query request to the feature query server. The feature query results are returned in GeoJSON used by the xGDS Share software. These GeoJSON query results are shown on the Share web display described earlier in this paper. Figure 5 shows an example of the approach used to integrate the feature query software with xGDS Share.

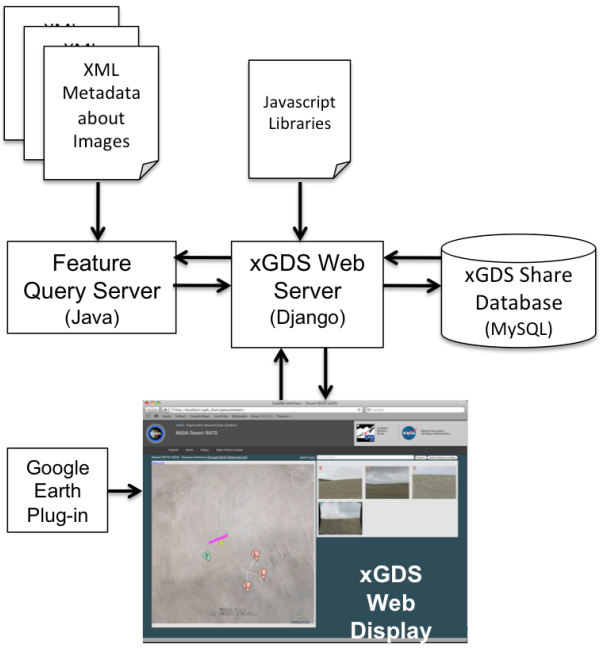


Figure 5. Integration of DELVE with xGDS

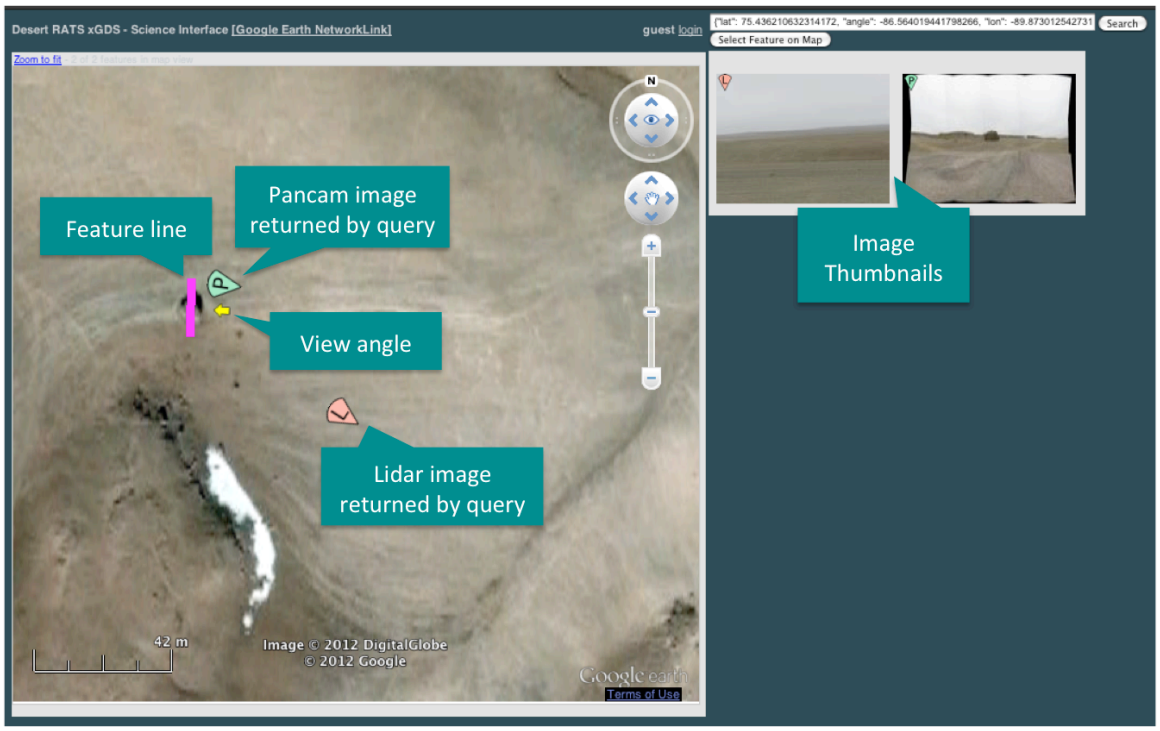


Figure 6. Example of xGDS Display of Feature Query Results

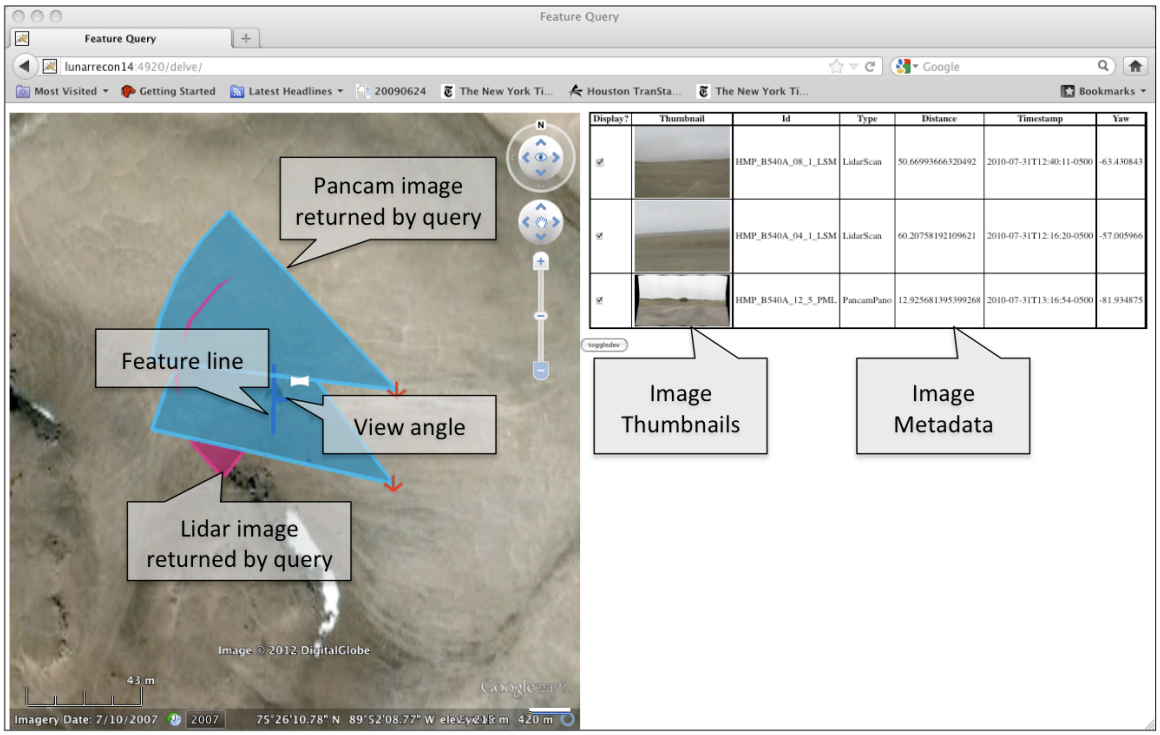


Figure 7. Example of DELVE Web Display of Feature Query Results

4. RELATED WORK

The feature query approach used in DELVE is a location-based search technique. Other such techniques include work by Shams et al. [5] on spatial queries for planetary data, and nearest neighbor approaches described in Zhang et al. [6]. For the spatial query, image datasets are grouped into spatial regions defined by adjacent rectangles and the R+ tree search method searches these regions to find images at a location. The user interacts with a map to identify a location on the rover's traverse path and all the images taken when the rover was at that location are returned. Like Sham's approach, we pre-compute spatial data structures for more efficient search. Our spatial structure, however, is the viewing cone of the image with its apex at the rover location. The viewing cone defines a spatial region that should be visible in the associated image and is unique to that image. Also similar to Sham's approach the user interacts with a map of the region to identify a location used to retrieve images.

There are some key differences between Sham's spatial query and the DELVE feature query approach, however. First the feature query algorithm accounts for the fact that the location associated with the image in the database is the location of the camera, not the locations visible to the camera. The feature query algorithm takes a feature location identified by the user and determines which images are likely to show that feature. It does this by identifying when the feature location is within the viewing cone of an image. Second the feature query algorithm searches for images taken from the user-specified viewing angle. Retrieving by location only can result in images facing away from the feature of interest.

The MSLICE (Mars Science Laboratory InterfaCE) is a data visualization, target selection, and activity planning software tool developed at NASA Ames Research Center and NASA Jet Propulsion Lab (JPL) for the MSL [2]. Like the feature query software, MSLICE includes software for visualizing the data acquired by a planetary rover. The intended use of MSLICE differs from the feature query software, however. MSLICE is used to identify where the rover should collect images and what kind of images to take, while DELVE is used to find data for analysis after it's collected. Another difference is that rover pose information used for MSLICE is based on dead reckoning and is less accurate than pose from the K10 robot. Thus spatial techniques for image indexing are less accurate for MSLICE. As a result, MSLICE users rely on additional information in user annotation to help identify targets in images.

5. CONCLUSIONS AND FUTURE WORK

The DELVE software has been used successfully for location-based search of the images collected by the K10 rover at HMP 2010. This demonstrates the potential of the feature query approach to support retrieval of images collected by a planetary rover. The DELVE software can be used with images collected during future rover operations by creating XML metadata files for these images. If these operations include new types of images, new metadata will need to be defined for these image types and the search algorithm will need to be extended.

We are adding new search capability to DELVE other than location-based search. We are investigating an approach to model metadata for images as semantic concepts that can be used to browse the image database. We have defined concept classes for sample collection by a planetary rover in the Web Ontology Language (OWL). We use the Hermit ontology reasoner (www.hermit-reasoner.com) to derive instances of the OWL classes (called *Objects*) from the XML metadata file. Attributes of Objects are stored in *Data Properties* e.g., Panorama *hasHeading* 30 degrees. *Object Properties* relate Object instances of Classes e.g., a Panorama *isSampleOf* some Task. Object properties are constrained by logical relations such as symmetry or transitivity. Rules defined in the Semantic Web Rule Language (SWRL) are used to infer relations.

We developed an example of this semantic metadata for the images collected at the HMP field test in 2010. We used the open source Protégé ontology editor (protege.stanford.edu) to manually encode Objects that define static concepts not in the telemetry, such as the Field Test name or the Rover identifier. We used the Hermit reasoner in Protégé to automatically build Objects that can be derived from information in the XML metadata files. This initial ontology is partial and was developed to demonstrate the feasibility of our approach. We plan to develop a full semantic model for images collected at HMP 2010, and use this model in the DELVE web application for semantic browsing of the Share image database.

6. ACKNOWLEDGEMENTS

This work was funded by a NASA Phase II Small Business Technology Transfer (STTR) contract. We would like to thank Maria Bualat, the Contracting Officer's Technical Representative (COTR) of this project, and Matthew Deans, the Deputy Group Lead of the Intelligent Robotics Group, for their support and advice on the DELVE project.

7. REFERENCES

1. Fong, T.; M. Bualat; M. C. Deans; B. Adams; M. Allan; M. Altobelli; X. Bouyssounouse; T. Cohen; L. Fluckiger; J. Garber; E. Palmer; E. Heggy; M. Helper; K. V. Hodges; J. M. Hurtado, Jr.; F.

Jurgens; T. Kennedy; L. Kobayashi; R. Landis; P. Lee; S. Y. Lee; D. Lees; J. Lum; M. Lundy; T. Shin; T. Milam; E. Pacis; E. Park; L. Pedersen; D. Schreckenghost; T. Smith; V. To; H. Utz; D. Wheeler; & K. Young. (2010). Robotic follow-up for human exploration. AIAA-2010-8605. In *Proceedings of AIAA Space 2010*; Anaheim; CA.

2. Powell, M. W.; Shams, K. S.; Wallick, M. N.; Norris, J. S.; Joswig, J. C.; Crockett, T. M.; Fox, J. M.; Torres, R. J.; Kurien, J. A.; McCurdy, M. P.; Pyrzak, G.; Aghevli, A.; Bachmann, A. G. (2009). MSLICE Science Activity Planner for the Mars Science Laboratory Mission. *NASA Tech Briefs*; September 2009; pp. 51-52.

3. Schreckenghost, D.; T. Fong; T. Milam; H. Utz. Measuring Robot Performance in Real-time for NASA Robotic Reconnaissance Operations. *NIST PerMIS Workshop*. Sep 2009.

4. Schreckenghost; D.; T. Fong; T. Milam. (2010). Measuring Performance in Real-time during Remote Human-Robot Operations with Adjustable Autonomy. *IEEE Intelligent Systems*. Sept./Oct. 2010.

5. Shams; Khawaja S.; Crockett; Thomas M.; Powell; Mark W.; Joswig; Joseph C.; Fox; Jason M. (2011) Spatial Query for Planetary Data. *NASA Tech Briefs*; April 2011; pp 32-33.

6. Zhang; Jun; Manli Zhu; Dimitris Papadias; Yufei Tao; Dik Lun Lee. (2003). Location-based Spatial Queries. *ACM SIGMOD'2003*; June 9-12; San Diego; California; USA.