[BACK TO INDEX]

ARTISAN:

An Integrated Scene Mapping and Object Recognition System

V. Broz, O. Carmichael, S. Thayer, M. Hebert, J. Osborn

Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213

{vbroz, owenc, sthayer, hebert, oz}@ri.cmu.edu

http://www.frc.ri.cmu.edu/projects/artisan/

ABSTRACT

Integration of three-dimensional textured scene mapping and object recognition presents many opportunities for assisted automation. We present Artisan, a software package that synthesizes these elements to form a user-friendly whole. Artisan uses a variety of 3D sensors, including laser range scanners and stereo systems, to acquire both image and range data. Artisan automatically finds the transformations between data taken at multiple sensor viewpoints using matching algorithms. The data from these viewpoints are then merged together to form an integrated textured map of the entire scene. Other user or sensor input can also inserted into the scene.

Using object recognition with an expandable library of objects, Artisan can identify and locate simple and complex scene features. With this identity and transformation information, it is able to support many operations, including semi-automatic robotic teleoperation and navigation. After mapping and recognition, the identity, position, and orientation of the objects in the scene can be automatically transferred from the Artisan system into other software, including robotic teleoperation packages. Numerous opportunities for automation exist during the operations stage as a result of this increased world knowledge.

This work was performed under contracts DE-AR26-97FT34314 and DE-AC21-92MC29104 for the Department of Energy, Federal Energy Technology Center, Morgantown, West Virginia.

1. Introduction

The rich 3-D data generated by range sensors during robotic 3-D mapping is not, in its raw form, abstract enough to provide machine-based assistance in manipulation tasks. This type of assistance, which we call 'semi-automatic robotic teleoperation', is possible only when the system has knowledge of the type, position and orientation of the objects within a scene to be manipulated. The Artisan system integrates a versatile 3-D mapping system with a powerful suite of object recognition algorithms and a robust persistent storage mechanism to provide both the photorealistic maps and the meta-information needed to automate teleoperation tasks.

Artisan has drawn inspiration from many past mapping and object recognition efforts. Its most recent precursor is a system developed at Carnegie Mellon that recognizes objects within a single 2.5-D acquisition [5]. The basis for that system is a set of algorithms developed for object recognition using 'spin images' [4]. The work described in this paper builds on that system by incorporating persistent storage of recognitions and acquisitions, multiple acquisitions, mesh matching, and mesh merging. These additions are vital for photorealistic mapping tasks. A concurrent project, C-Map, inspired many of these additions [1]. Originally based on the MarsMap software used during NASA's Pathfinder mission [10], this system features an interface for mapping using the trinocular stereo head aboard the Pioneer robot [8]. This robot is designed to map the interior of the Chernobyl Nuclear Power Plant 's disabled reactor. Although the robot features a manipulator arm, the system does not include semi-automatic teleoperation; its mission is reconnaissance, not repair. Artisan integrates and extends ideas from these past approaches by creating an extensible, powerful, and user-friendly tool applicable to a wide variety of sensing and manipulation scenarios.

2. Mapping and Recognition Technology

Artisan takes advantage of many novel technologies to produce photorealistic maps and accurate recognition results. To deal with large physical workspaces, Artisan merges multiple narrow-field-of-view 2.5-D sensor acquisitions with each other. This merging process requires finding the relative positions of the individual views using a number of algorithms, then creating a final, integrated surface mesh. Artisan can then recognize a set of known objects from that integrated view.

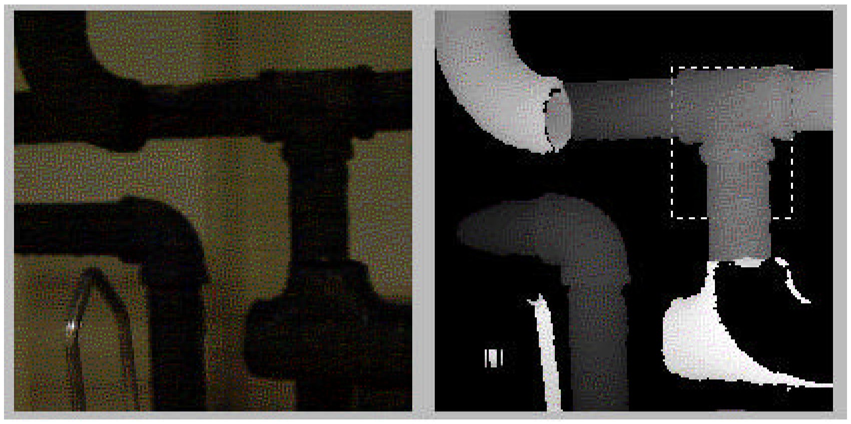

Figure 1: Camera image and range map--lighter pixels represent closer objects.

2.1 Image Acquisition

Artisan's internal format for range data is a simple array of x, y, z, u and v data representing a point-cloud (x, y, z) framed from a specific viewpoint (u, v). The u and v coordinates are associated with a corresponding image produced by the sensor, allowing texture to be added later in the process (see Figure 1). A specific sensor driver is written which transfers data from the sensor and rewrites it in Artisan's internal format. This may occur on the platform of the sensor itself, or within Artisan.

2.2 Mesh Creation

The initial 2.5-D representation is inadequate for the full 3-D data that is one of Artisan's end-products. Artisan uses Schewchuk's Triangle algorithm to produce constrained Delaunay triangular meshes from the range data produced by the range sensor [9]. An example of such a mesh is shown in Figure 2. This algorithm offers fast production of meshes with surface patches that guarantee a minimum angle constraint. This constraint eliminates 'slivers'--long, thin surface patches that may compromise the results of subsequent 3-D processing, including recognition and mesh matching algorithms.

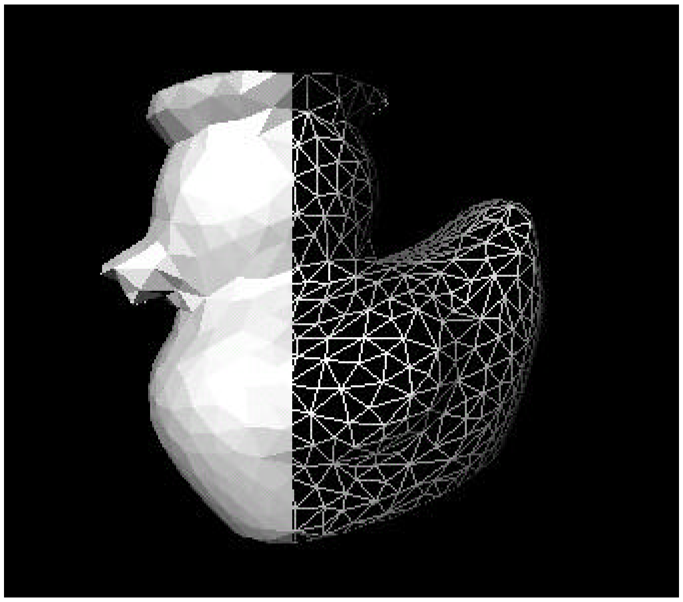

Figure 2: A surface mesh, shown partially in wireframe mode.

2.3 Mesh Matching

In many situations, the relative transformation between the position of the sensor during two different acquisitions is not easy to obtain. Mobile sensor platforms in unknown terrain very rarely feature precise orientation readings, much less accurate measurement of global position. This is often predicated by the very nature of mapping. However, accurate mapping requires that sensor readings have the ability to acquire and maintain ground-truth. In order to resolve this dilemma, methods have been produced to take advantage of the shared areas between two overlapping sensor acquisitions to find the most likely registration (relative transformation) between them. Two such algorithms, one known as SpinRecognize and the other as Iterative Closest Points (ICP), are used within Artisan to find such transformations [4][11].

When no estimate of the transformation aligning the two sets of sensor data is known, spin image matching is employed to acquire an initial, rough estimate of the transformation. In this algorithm, correspondences between points in one data set and points in another data set are found by comparing the local shapes of the surfaces surrounding those points. Shape comparisons are accomplished through the spin image [4]. Points with similar spin images are considered to have similar local shape. Once correspondences are found between the data sets, the transformation aligning the data sets may be estimated.

After the data sets are initially aligned, the ICP procedure refines the alignment. This algorithm iteratively minimizes the distance between points in one data set and the points in the other data set that are closest to them [11]. Within a few iterations, the two data sets are registered together with very little error. In order for ICP to work correctly, the data sets must be roughly aligned to begin with--otherwise, the correct match will not be found.

When the use of SpinRecognize and ICP is not possible, Artisan provides the user with a method of aligning the meshes manually using the mouse. After the user has roughly aligned the meshes, ICP can be used to obtain a more correct fit. As six degree of freedom manipulations are difficult to perform with a 2-D input device, we are looking into ways to use virtual reality hardware (possibly including data gloves) to provide a more effective manual manipulation method. Fortunately, we anticipate that manual mesh matching will only rarely be needed.

2.4 Object Recognition

Object recognition is a very similar problem to that of mesh matching. As such, a more general form of the spin image matching algorithm is used [3]. This algorithm can efficiently find multiple known objects within a given scene mesh, storing their position and orientation for future use. The spin image is designed to be resistant to clutter and occlusion--if part of an object in a scene is covered up, recognition is still possible. Using Principal Components Analysis (PCA), the spin images may be compressed, allowing recognition of objects from large libraries in a space- and time-efficient manner [2].

2.5 Mesh Integration

The construction of seamless, textured surfaces from an arbitrary number of co-registered surfaces meshes requires both a robust and efficient mechanism for accumulating support evidence in an incremental fashion, and robust operators that extract uniformly sampled surfaces from them. Artisan uses a modified implementation of Johnson and Kang’s surface synthesis technology, based on 3-D occupancy grids [6]. Their method represents surfaces using a probabilistic model that encapsulates a sensor error model (stereo) and an artificial point spread function. Each point from a surface mesh, along with the surface normal for that point, is distributed into the occupancy grid using the appropriate models. This evidence, both the surface likelihood estimate as well as the surface normal estimate is accumulated for each point from each input surface mesh. Surface extraction is performed using robust ridge detection on the surface probabilities in the voxel space that forms an implicit surface using both the likelihood gradient and the surface normal. Polygonization is performed using the Marching Cubes algorithm [7].

Whereas Johnson’s algorithm provides the baseline technology for the creation of well-conditioned surfaces from support evidence, it does not lend itself to the construction of large-scale models from high-resolution data. Our implementation of incremental surface construction sports several features missing from the original, which appear below.

Persistent Occupancy Grids

Efficient incremental surface construction is possible only with persistent occupancy grids. Our implementation is composed of three distinct steps: (1) **Definition** of appropriate grids for the task (initialization), (2) **Distribution** of surfaces mesh data into these grids (run once per mesh), and (3) **Extraction** of surfaces from occupancy grids (output).

Paging and Caching

Mature occupancy grids occupy hundreds of megabytes. Memory management in large-scale surface constructions is, therefore, critical. Our implementation features the encapsulation of dynamic paging and caching as a function of available RAM. This forces an absolute limit on the amount of virtual memory allocated, which is independent of grid space requirements.

Texture and Surface Channels

Our implementation sports a dynamic system for the allocation of surface evidence and texture channels. Application use requirements may range for the need for single static surface and texture channels to individual channels for each input mesh. These 'use' requirements are defined during the occupancy grid definition and persist throughout distribution and extraction.

3. System Architecture

Artisan's software architecture accommodates the aforementioned process. It includes a number of components that communicate via a TCP/IP-based network. This modular arrangement was chosen to deal with future additions and changes and to facilitate remote use over industry-standard networks. A diagram of the major components is shown in Figure 3, and a description of each component follows.

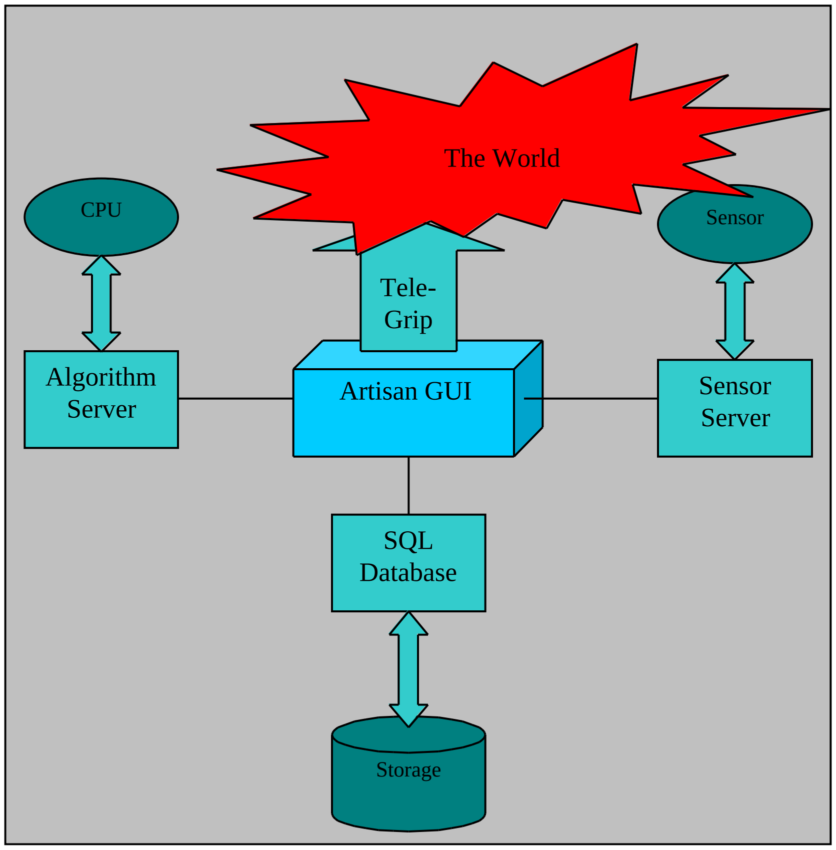

Figure 3: Artisan system architecture

3.1 Database

Artisan employs a persistent data storage model implemented in a local Structured Query Langurage (SQL) database. This database stores not only the end-results--the range data, corresponding images, original and merged meshes, and recognized objects--but also the parameters used in processing. This data is easily accessible via property dialog boxes. The robot operator can then make an initial assessment of the data, whereas an expert may later improve the result by reprocessing with modified parameters. For instance, the quality of the range data and the meshing parameters used often determine the quality of an automated matching of two meshes. In order to create a better integration based on that matching, an advanced user might either recreate the meshes from the original range images (stored in the database), or simply correct the misalignment by hand.

3.2 Algorithms Server

A client/server architecture allows Artisan to process data with the various algorithms used in its operation on a local or remote Algorithms Server. A command and the data to be processed are sent from the GUI component to the algorithms server. The server evaluates the request, processes it, and sends back the resulting data. The algorithms server currently runs on an SGI workstation. We originally found this combination of platforms for the GUI and algorithms server to be ideal--UNIX was exploited for its computational efficiency, where NT could be used to build a familiar and user-friendly interface. As Intel-based systems increase in power, however, we are considering porting this server to the NT platform.

3.3 Sensor Server

In order to integrate Artisan with off-the-shelf range sensors, we encapsulated the functionality of a range sensor in a platform-neutral TCP/IP-based server. A request for a sensor image, along with needed parameters, travels to the sensor in question, and the requested data (or an error message) is sent back from the sensor. To interface a new sensor into Artisan that conforms to its sensor abstraction model, a sensor driver is written. This driver transfers data from the sensor and rewrites it in Artisan's internal format. This rewriting may be done on the platform of the sensor, or within Artisan itself.

As a proof of concept, a new driver was written for system to interface with the Pioneer platform. Pioneer's sensor platform consists of a trinocular stereo system mounted on top of a pan/tilt unit aboard a mobile robot [8]. Code was written to interface with existing stereo algorithms, the sensor itself, and the pan/tilt unit on which the sensor was mounted. A subclass of the image class was created which encapsulated the additional information. Code in Artisan was written which allowed control of the pan/tilt unit and display of the pan/tilt position and sensor data. Radiation, temperature, and humidity data was available from the Pioneer's sensor platform. Accordingly, a new *Environmental Data* object was created in the GUI, along with code to interface with the environmental sensor. Because of the enabling infrastructure within Artisan, integrating the two systems took less than a week.

3.4 Graphical User Interface (GUI)

The GUI is the end-user's interface point with Artisan. It acts as a hub between the system components, requesting services and transferring data. Similar to the 'Explorer' tool in Windows, Artisan's GUI presents a hierarchical, object-oriented representation of its world.

Platform Choice

The GUI, shown in Figure 4, runs on a Microsoft Windows NT platform. This platform provides-interface familiarity to accommodate the 'non-expert' users to which the system is targeted. A standard array of buttons, lists, tree controls, and dialog boxes are available for use and familiar to anyone having previously used Windows 95 or Windows NT

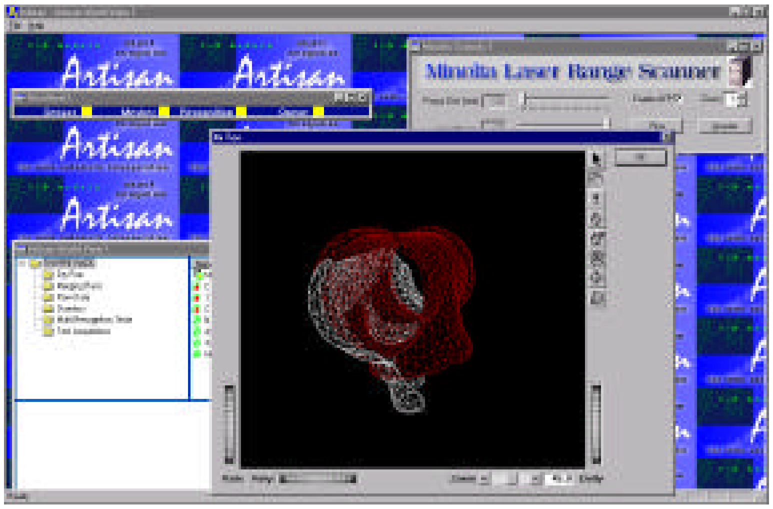

Figure 4: The Artisan GUI

Object-Oriented Representation

In Artisan, icons represent specific instances of different types of objects. Users can 'right-click' on any instance to view a list of operations they can perform on the object. The different types of objects that can be manipulated by the user are listed below.

Image Class

An *Image* object represents a single sensor acquisition. For the types of sensors used in our tests to date, this included a 2.5-D range image along with correlated camera image(s). The range image was translated into a textured mesh using the triangulation algorithm described above. As long as the data returned by a sensor can be transformed into a surface mesh, however, its initial format can be encapsulated in this class. By isolating the essential components of a 3-D sensor acquisition, Artisan facilitates the addition of new sensors.

Artisan does, however, enable special features of certain sensors to be used using subclassing and inheritance. For example, data from a sensor with a 'zoom' parameter would have a representation subclassed from the standard sensor. The meshing and merging algorithms, which do not depend on this data, would operate on the base class. A sensor-specific display module, however, would note the information.

Operations: View

The *Image* viewer allows the user to gauge the coverage and quality of the range image before transforming it into a surface mesh. At this time the user may choose to reject the range image, adjust parameters on the sensor itself, and try again. If the user wants to create a surface mesh from a subset of the full range image, he or she may select a region of interest and create such a mesh.

Create Mesh

This operation creates a textured mesh covering the full range image, as described above.

Subclasses: **_Minolta Image Class_**

A *Minolta Image* represents an acquisition from a Minolta Vivid 700 laser range scanner, consisting of a range image and an associated color image that acts as the texture for the mesh. It features the same operations as the *Image* class. The Minolta Vivid 700 was used during Artisan's creation.

Pioneer Image Class

A *Pioneer Image* represents the three images taken from the Pioneer trinocular stereo system, as well as the range image derived from them. It also features the same operations as the *Image* class.

Mesh Class

A *Mesh* object represents 3-D data contained in triangular surface mesh form.

Operations: View

The *View* operation brings up an OpenInventor-based window for the user to see the mesh on the screen. The mesh or the viewpoint can be moved to allow for a full view of the 3-D scene.

Subclasses: **_Single Mesh Class_**

A *Single Mesh* represents a *Mesh* that is continuous; contrast with a *MultiMesh*, which is formed by many discrete overlapping meshes.

Operations: Add to MultiMesh

This operation pops up a list of possible of *MultiMesh* objects in which the current *SingleMesh* can be inserted.

Recognize Objects

Using the recognition algorithm described above, the *Recognize Objects* command finds the most likely position and orientation of known objects within the scene. The user may select the objects to recognize via the dialog box shown in Figure 5.

Load into Telegrip

In order to actually manipulate objects within a scene, it is helpful to have unrecognized parts of the scene available during manipulation. This operation loads such a mesh into Deneb Telegrip, a product that we use as our manipulation tool. After it is loaded, the user can see that part of the scene while performing manipulation tasks.

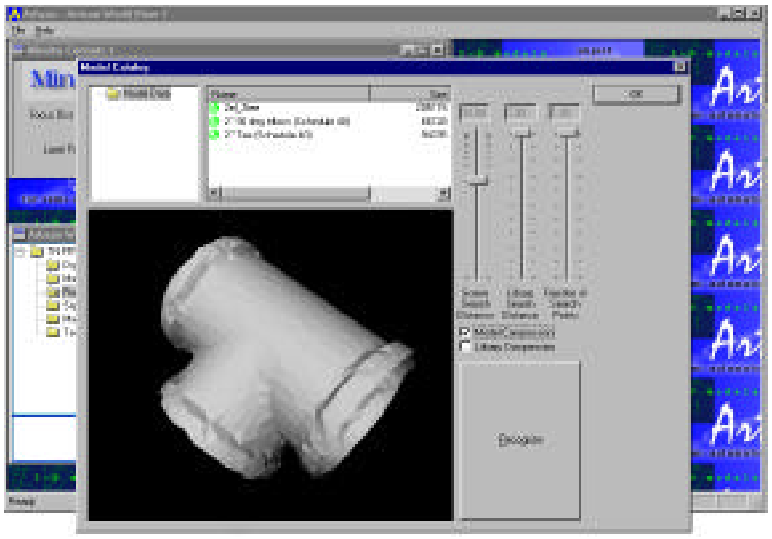

Figure 5: The recognition window

Subclasses: **Original Mesh Class**

An *Original Mesh* is one that was generated directly from an *Image* object--as such; it contains only a single textured surface mesh. It inherits the same operations as a *Single Mesh*.

Merged Mesh Class

A *Merged Mesh* represents a mesh developed using the surface integration technique described above. It is a single surface mesh, but instead of being created from an *Image* object, it is created from a *MultiMesh* object. (see below).

Model Class

A *Model* represents a 'known object' which Artisan can recognize within scene meshes.

MultiMesh Class

A *MultiMesh* consists of a number of distinct meshes in a tree arrangement, where each mesh is related to its parent via a transformation.

Operations: Merge

This operation uses the integration procedure described above to 'merge' the meshes within a multi-mesh to form a *Merged Mesh* object.

<u>Add Mesh</u>

This operation inserts a *Single Mesh* into the given *MultiMesh* by either using the SpinRecognize algorithm or manual placement by the user followed-up by Iterative Closest Points matching.

Recognized Model Class

A *Recognized Model* relates a *Single Mesh* with a *Model*--it stores the transformation determined by the Recognize Objects command shown above.

Operations: <u>Load into Telegrip</u>

As in the operation on the *Mesh* object above, this operation brings the *Model* in question into Telegrip.

<u>View</u>

This loads up a window showing both the recognized object along with the mesh from which it was recognized (Figure 6). The accuracy of the matching can be gauged from this view.

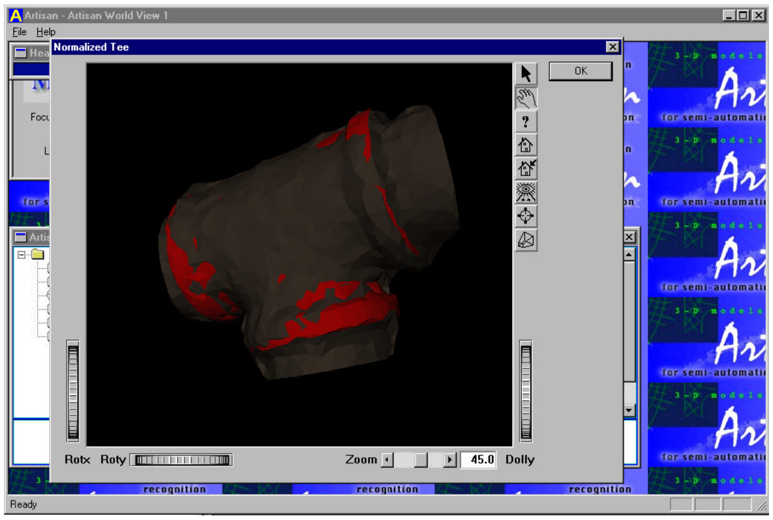

Figure 6: Model of tee (in red) recognized from a sensor acquisition.

Tree Representation

To support flexible use of the mapping data, Artisan requires a suitable representation of the information acquired and created. At the same time, however, ease-of-use needs restrict the choices available. A full-fledged scene graph, for instance, would be quite confusing for a novice user. On the other hand, a tree structure is a common paradigm on many personal computer platforms. Accordingly, the meshes, range and color images, and recognized objects which the user can view are held in a hierarchical system of folders. Different salient parts of a workcell can be mapped separately, avoiding the clutter of keeping all of the items in one folder. When a full map of the workcell is needed, the subparts can be matched and integrated into a complete workcell inside the parent folder.

4. Applications

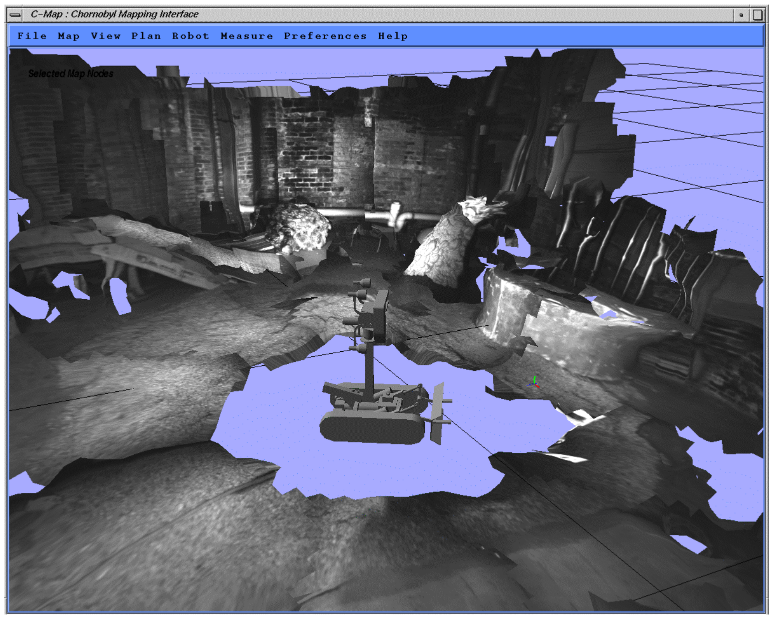
C-Map : Chornobyl Mapping Interface
File Map View Plan Robot Measure Preferences Help
Selected Map Nodes

Figure 7: A photorealistic world model, generated by C-Map [1]

Artisan can be applied to generic problems in both object recognition and the construction of 3-D photorealistic models from range data. Taken together, these capabilities serve important roles in the arena of semi-automatic teleoperation. The production of abstract, fully 3-D models via object recognition is essential to the rapid, accurate, and forceful interaction with the environment. For instance, manipulation tasks such as grasping, cutting, and material movement require a 3-D model of the world. With the addition of path planning, navigation through an unmodeled and potentially hazardous environment may proceed much more safely and with maximum efficiency.

5. Future Work

Artisan is currently a 'mock-up' scale utility. It does not scale to the modeling of large-scale, multiple-room scenarios such as complex nuclear power plants or waste-storage facilities. Operations on these orders of magnitude are critical for leveraging the full potential of the technology core that Artisan demonstrates. In the near future, we will equip the technological base with a domain knowledge capability. This knowledge, in the form of engineering constraints, will enable the precise recognition and registration in the face of sparse data, a problem that currently limits large-scale operations. In addition, a major systems engineering effort will be undertaken to build into Artisan the capability to filter, organize, and process the volumes of raw sensor date required to construct 'as-built' models of large industrial sites.

ACKNOWLEDGEMENTS

The authors would like to thank Herain Oberoi, Thomas Tadlock, and Scott Robbins for their past and present technical support. We would like to thank Andrew Johnson for providing many of the basic technologies that form the core of this work and David Kachmar for systems-level support. We would also like to thank Dr. William Hamel and his team at the University of Tennessee for their collaborations on the RTSA project, which funds this effort as a sub-contract. A special thanks to Vijay Kothari and the Morgantown Federal Energy Technology Center for their sponsor ship of the RTSA project.

REFERENCES

[1] T. T. Blackmon, L. Nguyen, C. F. Neveu, D. Rasmussen, E. Zbinden; M. Maimone, L.H. Matthies; S. Thayer, V. Broz, J. Teza, J. Osborn, M. Hebert, G. Thomas, J. Steele. "Photorealistic virtual reality mapping

system for Chernobyl accident site assessment." SPIE Photonics West 1999.

[2] A. Johnson and M. Hebert, "Efficient Multiple Model Recognition in Cluttered 3-D scenes." Proc. IEEE Conference on Computer Vision and Pattern Recognition. Santa Barbara, June 1998.

[3] A. Johnson and M. Hebert. "Recognizing objects by matching oriented points." Carnegie Mellon Robotics Institute Technical Report CMU-RI-TR-96-04, 1996.

[4] A. Johnson and M. Hebert. "Surface registration by matching oriented points." Proc. Int'l Conf. on 3-D Digital Imaging and Modeling (3DIM '97), Ottawa, Ontario, May 12-15, 1997.

[5] A. Johnson, R. Hoffman, J. Osborn and M. Hebert. "A system for semi-automatic modeling of complex environments." International Conference on Recent Advances in 3-D Digital Imaging and Modeling, (3DIM '97) pp. 213-220, Ottawa, Ontario, May 12-15, 1997.

[6] A. Johnson and S. B. Kang. "Registration and Integration of Textured 3-D Data." Cambridge Research Laboratory Technical Report CRL 96/4, September 1996.

[7] W. Lorensen and H. Cline. "Marching Cubes: A high-resolution 3-D surface construction algorithm." Computer Graphics (SIGGRAPH '87), pp. 163-169, 1987.

[8] Maimone, M., L. Matthies, J. Osborn, E. Rollins, J. Teza and S. Thayer, "A photo-realistic 3-D mapping system for extreme nuclear environments: Chornobyl," Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, October 13-17, 1998.

[9] J. Shewchuk, "Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator," First Workshop on Applied Computational Geometry, 124-133, May 1996.

[10] Stoker, C., Blackmon, T.T., Hagan, J. Kanefsky, B. and others (1997). MarsMap: Analyzing Pathfinder Data Using Virtual Reality. IN: 1997 Fall Meeting, American Geophysical Union, San Francisco, CA, Dec. 8-12 1997. vol. 78, no. 46. p. F403.

[11] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," Int'l J. Computer Vision, 13(2):119-152, 1994.