# Web-based Multimedia Development Techniques for the Instruction of Abstract Concepts in Computer Science

S. S. Iyengar
*Department of Computer Science*
*Louisiana State University*
*iyengar@bit.csc.lsu.edu*

Brian E. Pangburn
*Department of Computer Science*
*Louisiana State University*
*pangburn@bit.csc.lsu.edu*

Robert C. Mathews
*Department of Psychology*
*Louisiana State University*
*psmath@unix1.sncc.lsu.edu*

## Abstract

*Advances in information technology have provided educators with the tools needed to present abstract concepts in a clear and concise manner. These tools can be particularly effective in the field of computer science where the dynamic processes of algorithms must be conveyed to students at all course levels.*

*This paper presents a case study in the development of web-based tools for students learning about sensor systems. More specifically, an educational web site to supplement a textbook on sensor fusion is presented. The site uses Java visualization software and analogy based multimedia to reinforce concepts related to the text.*

*The site places particular emphasis on image registration, which is the process of finding the optimal mapping between two or more overlapping images. Image registration source code included with the text Multi-Sensor Fusion was rewritten in Java 2, numerous improvements were made, and a new registration technique known as TRUST was added.*

## Keywords

image registration, multimedia, sensor fusion, TRUST, visualization, Java, algorithm animation

## 1. Introduction

In the past few years, advances in information technology have provided educators with the tools needed to present abstract concepts in a clear and concise manner. These tools can be particularly effective in the field of computer science where the dynamic processes of algorithms must be conveyed to students at all course levels. In an introductory course, an instructor usually has to explain at least one sorting algorithm. In graduate level courses, algorithms are presented for a wide array of abstract concepts such as parallel computation, advanced data structures, global optimization, data compression and encryption, and computer vision. Historically, explaining any of these algorithms using only verbal description and static diagrams has often been tedious and difficult for even the best professors. This paper addresses some of these issues for the problems of image registration and global optimization through the use of visualization software and a multimedia enriched web site.

The web site stems from a project to develop visualizations of a new global optimization method known as TRUST that had been implemented in software for sensor systems. The visualizations were to be used for two primary purposes:

1. to help programmers better understand and evaluate the new global optimization method;
2. to teach those with little or no relevant background about both sensor systems and related software.

Once developed, the visualization software met the first requirement, but did little to help teach newcomers about sensor systems. Based on earlier work designing multimedia courseware for the Department of Psychology and the Louisiana Department of Transportation and Development (DOTD), several analogy-based multimedia animations were developed to help instill a basic understanding of the relevant concepts. It was hoped that after viewing the animations, the visualization software would be more appropriate for an inexperienced user.

Research shows that transfer is enhanced by instruction that helps students represent problems at a higher level of abstraction [2]. Benefits of helping students represent problems at a level of abstraction that transcends specifics of particular contexts have been demonstrated in many areas including algebra [14], computer language tasks [9], and analogical reasoning [7].

Upon completion of a prototype of the visualization software, it was decided that the work would be most beneficial if incorporated into a web site and used as a supplement to an existing textbook on sensor systems. Not only would it make the materials available to a wide audience, but also provide an excellent platform to add additional content including software instructions, related research materials, and a message board system for both educational and research collaboration [12].

We begin with some background on image registration, global optimization, and the TRUST method. Next, the development of the visualization software is presented in detail along with a technical discussion of system enhancements. This is followed by an overview of the creation of the multimedia content. Finally, the creation of the web site that incorporated the software and multimedia with a message board system is discussed.

## 2. Background

This project began with an effort to create a visualization system for image registration software based on source code included in the textbook, Multi-Sensor Fusion. Research was being done to extend some of the text's algorithms by implementing TRUST, a new global optimization method. Our initial effort was to visually compare TRUST with existing global optimization methods including genetic algorithms and taboo searches. This section will give a brief overview of image registration, global optimization, and TRUST [3].

### 2.1. Image registration

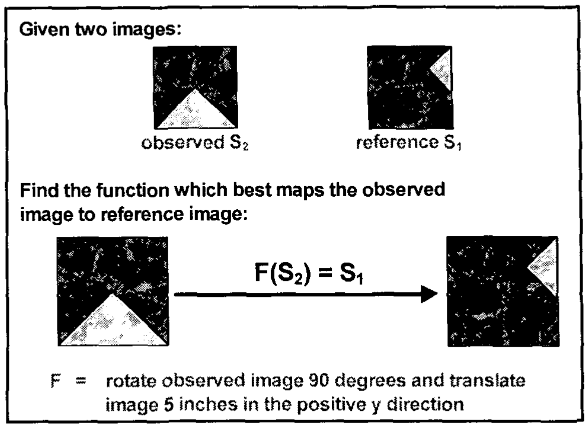
Image registration is essentially finding the mapping (or overlap) between two independent noisy sensor readings on a common coordinate system (Figure 1). For example, in a dynamic system where the sensors are in motion, readings may have to be merged without knowledge of sensor coordinates. This problem can be further complicated when noise is introduced.

For our project, it is assumed that all sensor readings are 255-shade grayscale images and are processed two at a time. It is also assumed that all images are two-dimensional and thus have horizontal, vertical, and rotational offsets referred to as X, Y, and θ respectively.

It has been shown that the ideal mapping between two images can be determined by finding the global minimum of the problem's fitness function. Given two sensor readings, S1 and S2, and a mapping ($X_i$, $Y_i$, $\theta_i$), each pixel, S2(x, y) in S2, is translated to a new position S2(x', y') and compared to S1(x, y). The fitness function is the sum of the squares of the differences in pixel values divided by the square of the total number of intersection pixels, yielding the equation:

$$\mathbf{f = [S1(x, y) - S2(x', y')]^2 / [\# \text{ pixels in intersection}]^2}$$

By incorporating the number of pixels in the intersection, mappings with higher overlap are favored. If two sensors overlap perfectly, the numerator of the fitness function will be zero. If the terrain from which the readings are taken has repeated characteristics there may be many "good" solutions but only one optimal solution. Thus, this approach to image registration can be classified as a global optimization problem [3].



Given two images:
observed $S_2$
reference $S_1$
Find the function which best maps the observed image to reference image:
$F(S_2) = S_1$
F = rotate observed image 90 degrees and translate image 5 inches in the positive y direction

Figure 1. Image registration problem

### 2.2. Global optimization

Global optimization is the process of finding the absolute maximum or absolute minimum of a multivariate function known as an objective function. The problem is that objective functions can contain many local minima and maxima (Figure 2). Optimization techniques can get "stuck" at these local optima. Global optimization research attempts to find ways to move beyond local optima to the global solution without using an exhaustive search. Since multivariate functions are used in many disciplines, global optimization techniques form an important category of research [12].

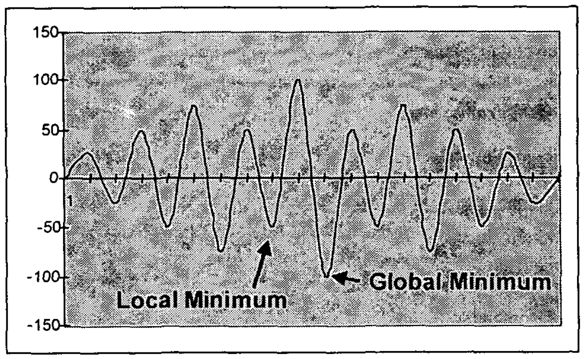


Figure 2. Global optimization

## 2.3 TRUST

The TRUST (*T*erminal *R*epeller *U*nconstrained *S*ubenergy *T*unneling) algorithm for global optimization was outlined in a 1997 article by Jacob Barhen et al. [1]. This new algorithm was a strong candidate for many global optimization problems because it was fast, it converged well, and it had clear, well-defined stopping criteria. TRUST's objective function combined sub-energy tunneling

$$Esub\ (x,\overset{*}{x}) = \log(1/[1+\exp^{(-(f(x)-f(\overset{*}{x}))-a)}])$$

with terminal repellers

$$Erep\ (x,\overset{*}{x}) = -(3/4)\rho(x-\overset{*}{x})^{4/3}\theta[f(x)-f(\overset{*}{x})]$$

where

**$a$ = constant related to the asymptotic behavior**

**$\overset{*}{x}$ = fixed value of x used to position the repeller**

**$\rho$ = strength of the repeller**

**$\theta$ = Heaviside function**

TRUST has been proven to converge in the one-dimensional case, but no proof yet exists for the multi-dimensional case [1].

## 3. Visualization software

### 3.1. Prototype

In 1998, Ying Chen implemented the TRUST method of global optimization for the image registration problem [4]. When applied to image registration, TRUST performed extremely well in the one-dimensional case. For benchmark testing, rotation was chosen as the variable parameter meaning that each sensor position was fixed both horizontally and vertically. Even with noise levels as high as 60%, TRUST found the correct rotation to within +/- one degree. In the multi-dimensional case, where rotation, x-offset, and y-offset all varied, TRUST usually came close to the correct values, but was rather sensitive to changes in function parameters [4].

Based on the encouraging results of Chen's research, it was decided to find a way to animate the various methods for image registration. The goal was to create a tool that would aid in visualizing the image registration process, and improve understanding of how TRUST compared with existing methods including genetic algorithms and taboo searches. As a secondary goal, the software was to be used to demonstrate the power of the various image registration techniques to others without requiring previous knowledge about sensor systems.

Animating the TRUST implementation for image registration raised some interesting issues. There were basically two approaches–either attempt to animate the internal workings of the TRUST algorithm, or animate its results with respect to image registration. While animating the actual algorithm might have provided some useful insights into TRUST, it would not have helped in evaluating its performance for the current application. It was decided to create a system that would display both the sensor images and the terrain from which they were taken, and then show the path that the algorithm took to find the optimal mapping. By animating these intermediate results, users could visually evaluate performance benchmarks including coverage, speed, and correctness. In addition, the system would be easily extendable for animating other optimization techniques. This would allow users to visually compare different algorithms and distinguish patterns and behavior that were particular to each [12].

Chen's original algorithm was written in ANSI C and was based on the routines provided in Multi-Sensor Fusion. When the program was run, the user would enter a series of parameters at the command line. After executing to completion, the routine created an ASCII data file for analysis. The easiest approach to animation would have been to run the program in batch, scan the data file for the intermediate coordinates of each sensor, and animate the sensor positions using a separate program. Unfortunately, this would not have made the program any easier to use since the parameters still had to be entered at the command line. More importantly, the batch approach would not have revealed anything about the speed of the algorithm at each step. It was decided to create a single program that would gather the input data using a graphical interface, display the initial sensor positions and images, and then animate the search in real-time.

A rapid prototype was developed in Visual Basic 5.0. Visual Basic was chosen to construct the prototype because it allowed for the quick and easy creation of the graphical interface. Screen components could be readily moved around the screen without modifying the source code.

Figure 3 shows a screen snapshot of the prototype. At the top of the screen are two fields that allow the user to generate a terrain internally or specify an external GIF, JPEG, or BMP graphics file. Beneath those fields is the picture box where the current terrain is displayed along with a graphical representation of each sensor. To the right of the terrain are six fields where the user enters the X and Y coordinates and the rotation for each sensor. Under the sensor fields is a button that the user can press to display the sensor extracts and begin the image registration algorithm. There is also a field to display the current iteration of the algorithm. At the bottom of the screen are fields to select which image registration method to use (TRUST was the only choice for the prototype) and the variance level (0 to 100) for the sensors.

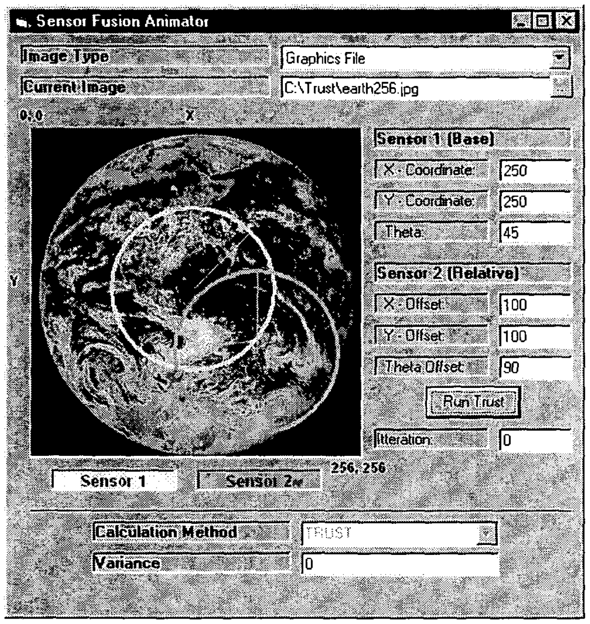


Image Type: Graphics File
Current Image: C:\Trust\earth256.jpg
Sensor 1 (Base)
X Coordinate: 250
Y Coordinate: 250
Theta: 45
Sensor 2 (Relative)
X Offset: 100
Y Offset: 100
Theta Offset: 90
Iteration: 0
Calculation Method: TRUST
Variance: 0

Figure 3. Visual Basic animation prototype

The prototype system proved to be useful almost immediately. During one of the first tests, it revealed an anomaly in the original source code that had been reversing the X and Y coordinates. Further tests showed that the system animated the image registration process well and allowed for visual evaluation of the terrain searches. For the first time, it was possible to see the search patterns in real-time and monitor the effect of parameter changes on those patterns. While useful, the prototype had several shortcomings including slower performance, platform dependence, and a number of calculation discrepancies that resulted from the C to Visual Basic conversion. In spite of these flaws, the prototype met its objective of demonstrating that a visualization of the sensors was useful, and that it would be worthwhile to move ahead with a more complete system.

## 3.2. Image registration applet

After we determined that the image registration animations were useful to developers, it became apparent that the same system could also be very helpful to students studying sensor fusion. Quite a bit of image registration source code had been provided for the applications in Multi-Sensor Fusion, but it had several potential shortcomings:

1. the code was very research-oriented and had been added to the text as an afterthought, primarily for informational purposes;
2. while there was considerable documentation in the code, it was probably not sufficient for a novice programmer;
3. although code was primarily written in ANSI C, students encountered compilation difficulties with some of the supporting libraries.

By carefully designing the new system, the above issues could be addressed, and a tool could be created that would encourage students to experiment with the algorithms. There was also the opportunity to better document the source code, and to standardize the interface between the subroutines so that new optimization methods would be easy to add. The new graphical interface would allow even non-programmers to run and test the programs. It was decided to port the existing ANSI C code to Java and deliver the application in the form of a web site applet. Converting the C code to Java would only require moderate changes and would result in a fully cross-platform system [6].

## 3.3. Porting C to Java

Since the original code was written in C and not C++, all the subroutines had to be converted to a series of classes. While difficult at first, the application mapped rather well to the object paradigm. Code related to the image that the sensors were extracted from was placed in a `Terrain` class, code related to extracting and storing the individual sensor images was placed in a `Sensor` class, and code related to each of the calculation methods was placed in the `Genetic`, `Trust`, and `Taboo` classes respectively. All of the applet initialization, graphics, and interface logic was placed in a `Fusion` class. Several additional classes were created to perform additional functions or hold additional data structures not directly related to the primary classes.

Once the C data structures and subroutines were mapped to objects, the actual conversion of the ANSI C code to Java was fairly straightforward. The two biggest tasks were removing the pointer references and replacing the function calls with method invocations. Since the data storage requirements were essentially constant throughout the applet, all of the pointers were replaced with arrays. Although linked-lists are supported in Java via references and would have worked equally well, we decided that arrays would probably be easier for a novice programmer to understand. To handle the use of methods instead of functions, the following format was used:

```
  result := Function_X(param list);
```

was translated to

```
  result := Object_Y.Method_X(param list);
```

where **Object_Y** was an instance of the class where **Function_X** had been rewritten as **Method_X**.

Porting the code also provided the opportunity to add quite a bit of additional documentation. The code needed to be fairly self-explanatory and readable by even a novice programmer. Every effort was made to remove redundancies and rename variables in an intuitive manner.

From a software engineering perspective, the object oriented redesign brought with it higher cohesion and lower coupling, adding to the overall maintainability of the code. Upon completion of the conversion, quite a bit of testing was done to insure that the ported algorithms produced results in line with the original ANSI C code. The graphical interface was a tremendous aid in this process. It allowed us to discover problems related to the parameter settings and the coordinate system quickly without having to scour entire data files of intermediate results [12, 13].

## 3.4. Graphical interface

Because Java is platform independent, it uses a different approach to create user interfaces. In the prototype Visual Basic system, controls (textboxes, pictureboxes, buttons, etc.) were placed in exact pixel locations in each window. In Java, controls have to be placed in approximate positions so that they can be rearranged at runtime based on the user's console type.

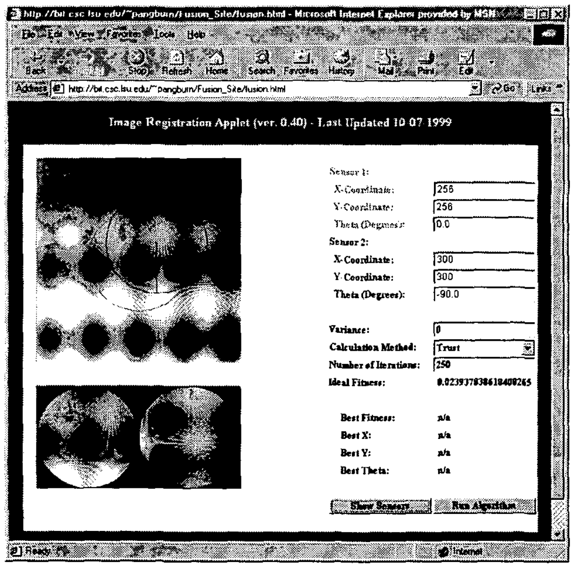

http://bit.csc.lsu.edu/~pangburn/Fusion_Site/fusion.html - Microsoft Internet Explorer provided by MSN
File Edit View Favorites Tools Help
Back Stop Refresh Home Search Favorites History Mail Print Edit
Address http://bit.csc.lsu.edu/~pangburn/Fusion_Site/fusion.html
Go Links
Show Sensors
Run Algorithm
Ready
Internet

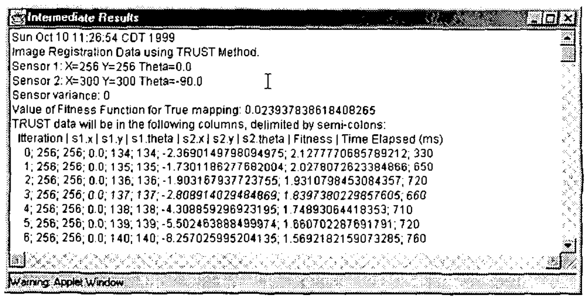
Image Registration Applet (ver. 0.40) - Last Updated 10 07 1999

Sensor 1:
X-Coordinate: 256
Y-Coordinate: 256
Theta (Degrees): 0.0
Sensor 2:
X-Coordinate: 300
Y-Coordinate: 300
Theta (Degrees): -90.0
Variance: 0
Calculation Method: Trust
Number of Iterations: 250
Ideal Fitness: 0.023937838618408265
Best Fitness: n/a
Best X: n/a
Best Y: n/a
Best Theta: n/a

**Figure 4. Image registration Java applet**

For the applet (Figure 4), the labels, textboxes, and buttons are grouped in a single panel object and arranged in a two by twenty-one (2 x 21) grid. By grouping the controls, they can be moved about the screen as a single entity while maintaining their relative position to one another. To the left of the grid are the two image areas. The large one displays the entire terrain, and the smaller one displays the two sensor images.

The user is provided with controls to set the positions of both sensors and variance (noise). There is a combobox for selecting the desired image registration method and a field to set the desired number of iterations for unattended runs. After the sensors have been extracted from the terrain, the ideal fitness is calculated using the "correct" mapping for the sensors. During the registration process, the "Best" fields are used to indicate the parameter values that have resulted in the lowest value for the fitness function. The **<Show Sensors>** button is used to extract the sensor images from the terrain based on the coordinates provided by the user. The **<Run Algorithm>** button is used to start the selected registration algorithm. During execution, the **<Show Sensors>** button is disabled, and the **<Run Algorithm>** button becomes the **<Stop Processing>** button, allowing the user to terminate the algorithm at any point.



```
Sun Oct 10 11:26:54 CDT 1999
Image Registration Data using TRUST Method.
Sensor 1: X=256 Y=256 Theta=0.0
Sensor 2: X=300 Y=300 Theta=-90.0
Sensor variance: 0
Value of Fitness Function for True mapping: 0.023937838618408265
TRUST data will be in the following columns, delimited by semi-colons:
 Itteration | s1.x | s1.y | s1.theta | s2.x | s2.y | s2.theta | Fitness | Time Elapsed (ms)
   0; 256; 256; 0.0; 134; 134; -2.3690149798094975; 2.1277770685789212; 330
   1; 256; 256; 0.0; 135; 135; -1.7301186277682004; 2.0278072623384866; 650
   2; 256; 256; 0.0; 136; 136; -1.903167937723755; 1.9310798453084357; 720
   3; 256; 256; 0.0; 137; 137; -2.808914029484869; 1.8397380229857605; 660
   4; 256; 256; 0.0; 138; 138; -4.308859296923195; 1.74893064418353; 710
   5; 256; 256; 0.0; 139; 139; -5.502463888499974; 1.660702287691791; 720
   6; 256; 256; 0.0; 140; 140; -8.257025995204135; 1.5692182159073285; 760
```

**Figure 5. Intermediate Results window**

Two major roadblocks were encountered while developing the Java interface–loading user terrains and storing numerical results. These roadblocks were caused by Java's strict applet security policy that prevents access to the local file system. By default, applets can only access files on the server from which they are downloaded. In our applet this resulted in the elimination of the feature to load a user-defined terrain. In order to allow the user to capture intermediate computational results we added an Intermediate Results window (Figure 5). Instead of writing results to a text file on the local file system, they are written to the user's display system using a separate window. The data is displayed in a semi-colon-delimited format with a header row containing field names. Users can cut-and-paste this data into a local text editor. Once saved from within the text editor, the resulting file can be loaded into a spreadsheet program or other numerical software for further analysis [11].

## 3.5. Technical highlights

During the code conversion, several enhancements were made to the entire software system. Several sections of the original ANSI C code were completely eliminated because the functionality they contained is built into the Java language. The most notable eliminations were the gaussian randomization routines that were replaced by Java's **`nextGaussian()`** method and the sorting routines that were replaced by Java's **`sort()`** method. We compared Java's sorting with two variations of the quicksort algorithm and discovered that the built-in method was significantly faster [13, 15].

Another feature of the Java language implemented for the project was support for multiple threads. The graphic processing and global optimization techniques are executed in a multithreaded fashion so that the user interface maintains responsiveness during periods of heavy calculation. As mentioned earlier, the user can press a button during the calculations to stop execution. This button press actually sends a message to the thread, telling it to destroy itself.

To handle the painting of the terrain and sensor extract images, Java's double buffering is utilized. Instead of drawing the grayscale pixels directly to the screen, each image is drawn to an off-screen image object. When complete, the entire image is copied to the screen. This prevents the user from witnessing the creation of the image. More importantly, it keeps the program from having to completely redraw the terrain and sensors each time the screen is repainted. Instead, the off-screen images are just recopied back onto the screen. This is significant considering that the screen is redrawn for each new sensor position displayed during the animation.

Quite a bit of work was done in all of the sensor manipulation classes to create a uniform coordinate system. As mentioned earlier, the Visual Basic prototype revealed that the X and Y coordinates were being reversed in certain places. For the applet, the entire coordinate system was revamped to make it more intuitive for inexperienced users. A "high school geometry-like" system was used with X increasing from left to right, Y increasing from bottom to top, and theta measured counter-clockwise. It is important to note that in the graphics routines, the Y coordinate is always reversed since the vertical component in Java (and many other languages) is measured from top to bottom.

One final area where several technical improvements were made was the TRUST algorithm. First, the search was modified to start in the lower-left quadrant of the terrain instead of in the middle of the terrain. Due to the nature of the fitness function, the ANSI C algorithm had been limited to searching the upper right-hand quadrant, leaving the other three untouched! To resolve another problem where TRUST only searched coordinates where X was equal to Y, the user was given an option to control the type casting in the partial derivative calculations. The last major calculation enhancement was to modify the criteria for TRUST's gradient descent phase. Originally, the algorithm used the square root of $(X^2 + Y^2 + \theta^2)$, but evaluated theta in radians. This lead to a disproportional weighting of theta in the evaluation, and allowed changes in theta to drive the algorithm. By converting theta back to degrees, the weighting was normalized and the unfavorable evaluation corrected [12].

## 4. Multimedia content

While the applet is useful for researchers and students with some background in sensor systems, it does little to help newcomers understand image registration. To bridge this gap, several multimedia animations were developed to explain the basic concept of image registration. These animations are based on a concurrent project in cognitive psychology to develop courseware for the Louisiana DOTD. This courseware is aimed at advancing the math skills of employees with limited education.

Two animations were created using Macromedia's Director. Both animations contain high-resolution graphics, sound effects, and complete verbal narration. These files were compressed for delivery via a web site using Director's Shockwave technology.

The first animation (Figure 6) is an elaboration of a real-world application of image registration that was presented in Multi-Sensor Fusion using diagrams. A graphic of the earth with an orbiting satellite is presented to the user. At a timed interval, a sensor on the satellite captures four overlapping images of the earth's surface. The four images are displayed, and the earth and satellite are moved to the corner of the screen. The four overlapping images are then reconnected to form a larger image. The narration explains that when computers are used to reconnect overlapping images, the process is called image registration [12].

The other animation (Figure 7) uses an analogy-based approach to present the image registration problem in more familiar terms. Since many viewing the site may not be familiar with satellite systems, the problem is mapped to the more common task of taking photographs with a camera. This analogical approach to creating the animation is based on the works of Holyoak and Thagard. They explain that analogy can be an effective tool in teaching unfamiliar scientific concepts to a novice [8]. In the field of computer science, Matocha et al. have successfully incorporated analogy with exaggeration in lectures introducing concepts related to computer networks [10].

The animation places the viewer in front of the Eiffel Tower with a camera. Initially, the entire tower is within the viewfinder. The narration explains that the photographer wants a close-up picture of the tower so he/she begins to close in on the tower. As the camera zooms in, it becomes apparent that it is impossible to capture the entire tower in a single photo from the desired distance. To solve this problem, two photographs are taken–one of the bottom half and one of the top. The narration explains that when the photographer later determines how to put the two separate pictures together, he/she is performing the same task as the various image registration algorithms.

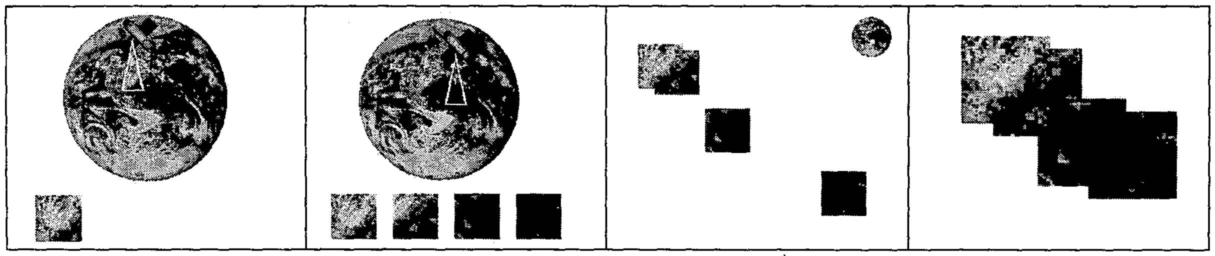


Figure 6. Orbiting satellite animation screen shots

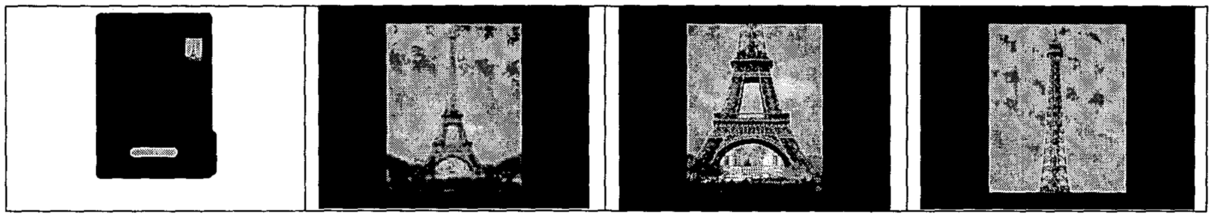


Figure 7. Image registration analogy: Eiffel Tower photos

## 5. Web site

In order to disseminate the image registration applet and the analogy based animations a web site was created. The site focuses on supplementing Multi-Sensor Fusion so that it can be used in conjunction with the text in a course on sensor systems. This allows the work to be integrated in a coherent form and provides an excellent medium to add additional content in the future. The address for the site is:
http://bit.csc.lsu.edu/~pangburn/Fusion_Site/textbook.html

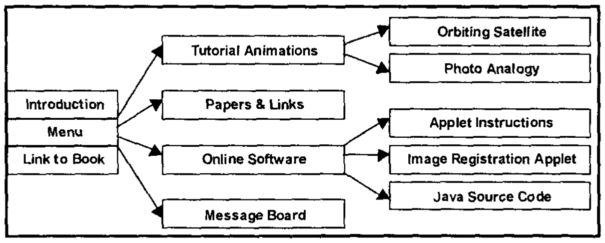


Figure 8. Map of the web site

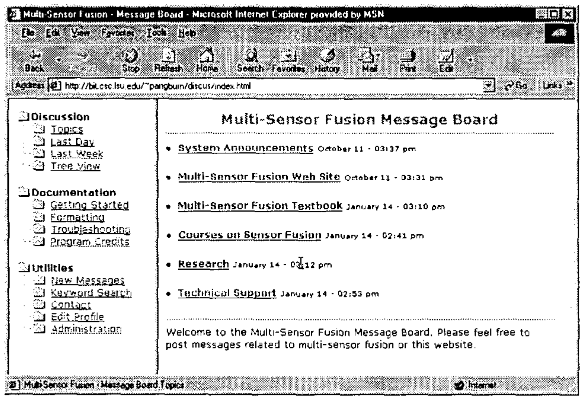
Figure 8 shows a chart of the web site's layout. The site begins with an introductory page containing a brief explanation of the site, a set of navigation links, and a link to obtain more information on the source textbook. The navigation links allow a visitor to explore the four different sections of the site–*Tutorial Animations; Papers & Links; Online Software; Message Board.*

The first section, *Tutorial Animations*, brings the user to a page containing the analogy-based animations of the satellite system and the Eiffel Tower. Basic instructions and a link to download the Shockwave viewer are provided.

The second section, *Papers & Links*, is used to disseminate current research related to the text by placing current research papers and links online. Any documents not in HTML have been converted to Adobe's PDF format since a free viewer is available for most computer platforms. A link to download the viewer is provided.

The third section, *Online Software*, contains instructions on the application software and links to both the Java applet and the complete source code. For those who choose to examine and/or modify the code, there are several links to Java programming resources on the Internet.

The final section, *Message Board*, (Figure 9) is a public message board system that can be used to provide feedback on the site and share ideas and research with other visitors. It is our hope that the board will be used by both educators and students. The board has been initialized with sections for chapters of the text, research, class notes and lecture materials, group projects, technical support, and miscellaneous discussions. The board is CGI-based and was created using Discus, which is a free discussion board software package available via http://www.chem.hope.edu/discus/.


Multi-Sensor Fusion - Message Board - Microsoft Internet Explorer provided by MSN
File Edit View Favorites Tools Help
Back Stop Refresh Home Search Favorites History Mail Print Edit
Address http://bit.csc.lsu.edu/~pangburn/discus/index.html Go Links
Multi-Sensor Fusion - Message Board Topics
Internet

Discussion
- Topics
- Last Day
- Last Week
- Tree View

Documentation
- Getting Started
- Formatting
- Troubleshooting
- Program Credits

Utilities
- New Messages
- Keyword Search
- Contact
- Edit Profile
- Administration

Multi-Sensor Fusion Message Board

- System Announcements October 11 - 03:37 pm
- Multi-Sensor Fusion Web Site October 11 - 03:31 pm
- Multi-Sensor Fusion Textbook January 14 - 03:10 pm
- Courses on Sensor Fusion January 14 - 02:41 pm
- Research January 14 - 03:12 pm
- Technical Support January 14 - 02:53 pm

Welcome to the Multi-Sensor Fusion Message Board. Please feel free to post messages related to multi-sensor fusion or this website.

Figure 9. Discus Message Board

## 6. Summary

We have presented the development of a Java 2–based visualization system for evaluating and testing image registration algorithms. The visualization system provides insight into the behavior of various image registration techniques and was instrumental in debugging and improving several of the methods. These improvements include correcting anomalies in the coordinate system, improving search space coverage, and minimizing local optima traps.

In order to make the code more understandable to novice programmers, additional documentation was added and variables were renamed in a more intuitive manner. Wherever possible, complex data structures were replaced with simpler structures. Several sections of code were entirely eliminated as a result of Java 2's extensive library of routines. Other system enhancements include replacing hard coded values with constants, removing redundant calculations, and converting ANSI C routines to Java's object paradigm.

A web site was created to disseminate the software and to serve as an educational supplement to the textbook, Multi-Sensor Fusion. Additional content including analogy-based animations and research links were added to enhance learning. A complete message board system was also provided for collaborative efforts.

It is hoped that this site and the visualization software will be used in conjunction with future courses on sensor fusion. In addition, the visualization software should be a very useful tool to both students and researchers who are evaluating existing image registration methods and developing new algorithms. The code was written so that it can be readily extended for other methods.

## 7. References

[1] J. Barhen, V. Protopopescu, and D. Reister, "TRUST: A Deterministic Algorithm for Global Optimization," *Science*, vol. 276 - May 16, 1997, pp. 1094-1097.

[2] J. D. Bransford, A. L. Brown, and R. R. Cocking. (Eds.), "How people learn: Brain, mind, experience, and school," *Special Report of the National Research Council*, National Academy Press, Washington D. C., 1999.

[3] R. R. Brooks and S. S. Iyengar, *Multi-Sensor Fusion: Fundamentals and Applications with Software*, Prentice Hall PTR, Upper Saddle River, NJ, 1998.

[4] Y. Chen, *Application of TRUST for Image Registration*, master's project report, Louisiana State University, Baton Rouge, LA, 1998.

[5] Z. Chen, *Implementation of TRUST in C++ and Its Application in Multidimensional Spaces*, masters project report, Louisiana State University, Baton Rouge, LA, 1999.

[6] S. R. Davis, *Learn Java Now*, Microsoft Press, Redmond, WA, 1996.

[7] M. L. Gick, K. J. Holyoak, "Schema induction and analogical transfer," *Cognitive Psychology*, vol. 15 - 1983, pp. 1-38.

[8] K. J. Holyoak and P. Thagard, *Mental Leaps: Analogy in Creative Thought*, MIT Press, Cambridge, MA, 1995.

[9] D. Klahr and S. M. Carver, "Cognitive objectives in a LOGO debugging curriculum: Instruction, learning, and transfer," *Cognitive Psychology*, vol. 20 - 1988, pp. 362-404.

[10] J. Matocha, T. Camp, and R. Hooper, "Extended Analogy: An Alternative Lecture Method," *Proceedings of the 29th SIGCSE Technical Symposium on Computer Science Education*, ACM, New York, NY, 1998.

[11] P. Naughton and H. Schildt, *Java 2 – The Complete Reference*, Osborne/McGraw-Hill, Berkeley, CA, 1999.

[12] B. Pangburn, *Development of an Information Technology Web Site to Supplement Multi-Sensor Fusion*, master's project report, Louisiana State University, Baton Rouge, LA 1999.

[13] S. R. Scrach, *Classical and Object-Oriented Software Engineering*, WCB/McGraw-Hill, Boston, MA, 1999.

[14] K. Singley and J. R. Anderson, *The transfer of cognitive skill*, Harvard University Press, Cambridge, MA, 1989.

[15] M. A. Weiss, *Data Structures & Algorithm Analysis in Java*, Addison-Wesley, Reading MA, 1999.