

Interactive Visual System

Arthur Evans, John Sikorski, and Patricia Thomas

Abstract

The Interactive Visual System (IVS) exploits the vast pattern recognition capabilities of humans and the computational power computers. The project grew out of work by Dr. George Nagy and his graduate student, Jie Zou, at Rensselaer Polytechnic Institute, who recently developed a system called CAVIAR (Computer Assisted Visual Interactive System). Running on a desktop, CAVIAR identifies a flower based on features that are interactively extracted from an image and submitted for comparison to a species database. IVS has similar functionality to that of CAVIAR. However, in addition to a desktop, IVS is designed to run on a Sharp Zaurus SL-5500 handheld computer, offering a greater degree of portability for practical and real-time use in the field. A k-nearest neighbor algorithm provides a robust pattern recognition technique using any number or combination of five different features for classification. Our findings indicate that IVS outperforms humans alone on speed and machines alone on accuracy.

1. Introduction

Over the past decades, researchers have sought with mixed success to have computers attain or surpass the power of the human mind. Many techniques have been developed to provide computers with more human-like characteristics such as genetic algorithms, dynamic programming, and artificial intelligence, but none seem to have come close to replicating the full perceptive powers of the human mind.

A human is able to quickly and accurately discern an object from its surroundings, search through his/her memory base of knowledge and determine the object's identity. How the brain manages such large-scale and instant pattern recognition is still not well understood. A shortcoming of the human brain, however, is that humans are limited to objects they have previously encountered or perhaps objects that have been described to them. Also, humans generally learn by a conventional, relatively slow means.

Computers, on the other hand, are capable of extremely large-scale memory storage and recall, fast computational skill, and high reproducibility. Disk

arrays provide a near limitless amount of memory from which a computer can draw. Computers are capable of performing complex mathematics almost instantly with no variability between multiple executions of the same task. Programming languages provide an interface to the computer's "mind," allowing us to make a computer do what we want. Still, with all this artificial mental power, computers have one shortcoming, they cannot think, but instead must follow a strict set of instructions set forth by lines of programming code.

This study employs IVS to examine the pronounced differences between humans' analytical skills and the computational power of machines. Our findings indicate that IVS outperforms humans alone on speed and machines alone on accuracy.

2. System Architecture

2.1 Hardware

At this project's inception, small portable computers fell into two categories: personal digital assistants (PDA's) and handheld computers. PDA's typically have slow processors and limited memory.

They excel at applications such as address books and calendars but do not have the memory or processing power we require for this application. Handheld computers, on the other hand, have significant processing power. At project inception, an average handheld computer had a 200MHz processor and about 64 MB memory, examples being the Compaq Ipaq series and the Sharp Zaurus.

Our choice was the Sharp Zaurus SL-5500 equipped with a 200Mhz processor, 64 MB RAM, and Compact Flash and SD ports. Running a Linux operating system by Embedix [13] and Java support through Personal Java, the Zaurus was unique in the field. With a relatively low cost, about 30% less than similarly equipped competitors, the combination of Java and Linux made for an inexpensive yet powerful system.

Images for identification are loaded to the handheld by two different methods. Using the Zaurus' Compact Flash port, an image can be captured and uploaded by most digital cameras. A second means is with the optional camera attachment by Sharp (CE-AG06). Inserted into the Compact Flash port, the camera attachment allows the user to capture an image directly from the handheld computer.

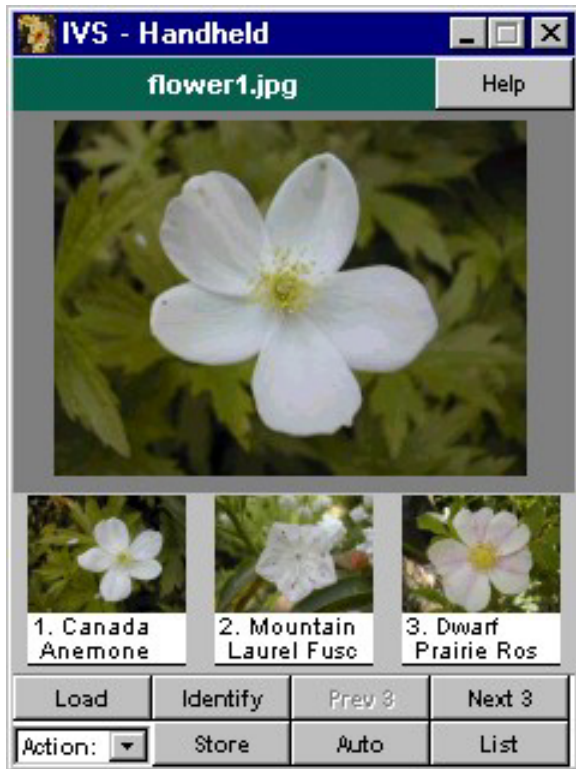


Figure 1 – IVS User Interface.

2.2 Personal Java vs. J2ME

Java 2 Micro Edition (J2ME)[11] consists of Java implementations targeting small devices from Smart Cards to high-end handheld computers. Sun designed J2ME with the concept of Configurations and Profiles. A Configuration targets a class of devices and is tied closely, on a low level, to the implementation of the Java Virtual Machine (JVM). Criteria such as memory, display, connectivity and processing power are taken into consideration. Currently, two configurations are well defined: Connected Limited Device Configuration (CLDC) and Connected Device Configuration (CDC). CLDC is the most basic Configuration requiring a total of only 128 KB memory with 32 KB available for the JVM, whereas CDC requires a total of 512 KB memory and 256 KB for the JVM.

Profiles were developed to sit on top of Configurations, providing the libraries required by programmers to write applications for a class of devices. There are two available profiles: Mobile Information Device Profile (MIDP) and Personal Profile. MIDP provides limited functionality for user input, networking and persistent storage, and has a very basic GUI toolkit. MIDP is typically implemented on PDA's, pagers and cellular phones. Personal Profile targets high-end devices, such as the Zaurus, providing a complete set of libraries available in the Java 2 Standard Edition. Personal

Profile is a fairly new specification and is not yet widely used. A reference implementation for the Zaurus was provided by Sun[11], but unfortunately, due unavoidable bugs, it could not be used for our IVS application. All IVS code was written to Personal Java specifications.

Personal Java predates J2ME, so it does not implement any Configurations or Profiles, although it is most similar to Personal Profile. Since our code was written to Java 1.1.8, when Personal Profile becomes an option for this and other handhelds, our code should run without issue.

2.3 Software

IVS was designed with enough abstraction to allow for easy code migration to other object identification systems. The code consists primarily of two packages, the GUI and K-Nearest Neighbor classifier (KNN). All classification code is fully abstracted into generic and abstract classes. Two interfaces provide the framework for object type-specific data—one for data sent from the GUI and another for database information.

2.4 Database

For simplicity, speed and portability we chose to save data as a simple tab-delimited text file. Each instance is stored as one line in the file. All data is stored in its raw (pre-normalized) form along with its species name and a reference file name. Images are stored in an “images” directory with subdirectories for each species.

3. User Interface

User interaction is a defining feature of the IVS. Interaction with a handheld computer is similar to that of a desktop. Instead of using a mouse, the user interacts with the application using a pen-like stylus. With a limited screen size, 240x280 pixels, display features must be optimized to fit in a relatively small space. The IVS graphical interface was built using standard Java AWT components (**Figure 1**).

A ‘Load’ button allows the user to load an image from the handheld file system, or from an inserted Compact Flash or SD memory card. The IVS resizes and centers the target image into a 240x150 area, while maintaining its original aspect ratio. Space is reserved immediately below the target for thumbnail images, 80x50 pixels each, which are returned from the classification engine after feature extraction.

The bottom panel contains a drop-down menu to select a feature action, an ‘Identify’ button, and a ‘Store’ button. Each feature action has an accompanying toggled panel displayed at the top of the screen. A ‘Help’ button provides useful information for each action. Once the user performs the action, the ‘OK’ button submits the data for storage.

3.1 Interactive Feature Extraction

Through simple interactions such as pointing and dragging, the user indicates a number of distinct flower features. The user may choose up to five features: primary petal color, secondary petal color (if present), stamen color, petal count, and petal/flower aspect ratio. For the purpose of our study, however, we focus simply on the three color-related features.

3.2 Colors

Pressing to a region of interest with the stylus will record pixel color. The user verifies correct color selection by a small colored rectangle in the top-most panel. The classifier handles each color internally as three separate byte values representing red, green, and blue channels, respectively. In the case of two-tone petals, the user selects the predominant color as the primary petal color (Figure 2).



Figure 2 – Features are extracted.

3.3 Automatic Feature Extraction

As an alternative to interactive feature extraction, the user may extract the three dominant color features automatically by clicking the 'Auto' button. IVS segments the flower using Sobel edge detection, followed by line thinning and noise reduction algorithms. The flower shape is then derived as an array of integers, where a center is estimated from the

average point of the detected edge pixels, and rays are drawn from center to edge at one-degree intervals. Edge gaps are "filled" by repeating the length of the last ray to reach an edge. The target is then stored as a polygon, with each ray endpoint comprising one of 360 vertices. Points that lie within the polygon are considered object foreground, while those that lie without are discarded as background.

Once the target flower has been segmented, primary and secondary petal colors are determined by the first and second most frequent pixel colors lying within the target region. The center or stamen color is taken from the center pixel previously derived.

3.4 Identification Process

After the user has completed feature extraction, he/she clicks the 'Identify' button. Reference thumbnails and species names for the three closest matches are displayed below the image. By clicking on a reference image, it is enlarged to the target image size. If the user positively identifies the species, its data can be added to the database by clicking the 'Store' button, which may also be used to store new species.

3.5 Classification

Classification is accomplished using the k-nearest neighbor algorithm. The classification engine accepts any number or combination of features and calculates Euclidean distance to all instances in the database.

$$\text{Distance} = \sqrt{(A_x - B_x)^2 + (A_y - B_y)^2 + \dots}$$

All features are normalized and unweighted. The classification engine returns the top three nearest neighbors by species. With a value of k=1 the nearest neighbor is the species with the closest instance in Euclidean space. The second and third nearest neighbors are the second and third species encountered while ascending the distances vector. Once a positive identification is made, instance data can be added to the database (Figure 5).

4. Evaluation

In order to test the effectiveness of the IVS system we designed some preliminary tests to determine classification times under different conditions. Our intent was to see how speed and accuracy is enhanced using IVS.

4.1 Test Systems

Testing on the handheld was performed on a Sharp Zaurus SL-5500 with 200MHz Intel StrongArm processor. The desktop system was a Dell Optiplex GX240 with a 2000 MHz Pentium IV processor. The

field guide was a National Audubon Society Field Guide to North American Wildflowers [6].

4.2 Design

We had three users classify 20 flowers under five different conditions:

- 1) Using IVS interactively on the handheld.
- 2) Using IVS interactively on a desktop.
- 3) Using IVS with automatic feature extraction on the handheld.
- 4) Using IVS with automatic feature extraction on the desktop.
- 5) Searching through a field guide.

For tests 1-4, we recorded times for feature extraction alone, classification, and total time. Percent accuracy was determined for the first hit being correct and the correct species in the top three hits.

4.3 Results

We present our results as observations, not necessarily statistically significant data (Figures 3 and 4). In order to scientifically test the system we would require a significantly larger database and a larger user sampling. Our observations show IVS greatly enhances speed and accuracy over computer or human alone classification.

Since handheld processing power is significantly less than that of a desktop, we recorded results with IVS on a desktop as a system comparison. Having a faster processor and possibly a more optimized JVM, the desktop results were faster than those on the

handheld. As expected, accuracies for both systems are similar.

The most common method used by a layman to identify a flower is searching through a field guide. Our observations show an average time of about two minutes fifteen seconds to correctly identify a flower. Automatic feature extraction and classification is significantly faster at 26.4 seconds but accuracy is only 65%.

When IVS is used on the handheld, accuracy jumps to 90% with an average time of 26.7 seconds. Interactive accuracy is less on the desktop at 83%, a difference most likely attributed to user variation in feature selection. A larger user sampling would likely pull these numbers closer together.

IVS is a computationally intensive application, pushing the handheld system to its limits. Overall computational time on the handheld is more than an order of magnitude slower than on the desktop. A majority of the classification time is resizing images for display to the user. Since user interaction times are almost the same on both systems, as handheld processor speed increases, classification times should significantly decrease.

5. Conclusion

We designed the IVS with extensibility in mind. Although written to identify flowers, a framework was created that can be utilized to identify any number of objects. The recognition engine is fully abstracted, requiring the implementation of a few interfaces for data handling. Due to its object-specific nature, GUI modification would require some work, but tools for handling all necessary image manipulation are present. With a minor amount of additional effort, the IVS framework could be extended to identify any number of objects. Potential uses include identification of objects in fields such as botany, engineering, law enforcement and medicine.

Identifying a flower is not a particularly difficult task for a human. Numerous field guides describe every known species in all geographic regions with cross-referenced indexes to aide identification. The problem is time. As our observations show, the IVS application significantly decreases the time required to identify a species compared to using reference books. A computer alone is certainly faster at this task, but our observations also show human interaction-specifically in carrying out the task of feature extraction-increases the accuracy of species identification.

This application was written as a proof of concept, showing human interaction with an image on a handheld and porting a desktop application design to a handheld. Although we successfully accomplished these tasks, we see much room for further research and development. Two main areas are: how to increase accuracy and how to increase speed.

Time (sec):	Handheld	
	Machine	Interactive
Extract	14.5	14.8
Classify	11.9	12.0
Total	26.4	26.7
Accuracy:		
% Correct	30%	63%
% Top 3	65%	90%

Figure 3 – Classification times using handheld.

Time (sec):	Desktop	
	Machine	Interactive
Extract	0.4	14.3
Classify	0.5	0.5
Total	0.9	14.8
Accuracy:		
% Correct	30%	63%
% Top 3	65%	83%

Figure 4 – Classification times using desktop.

Our species database is far too limited to be useful in any real-world application. As the database grows, identification accuracy will likely diminish. Our observations show color as a more important distinguishing feature, but a larger database will result in significantly more species with the same or similar colors. Other features must be used. Our application was designed to identify flowers using features extracted from an image, but future development should include additional features, some of which may not be extractable from the image but can be measured and entered by the user. Suggestions from botany domain experts would shed light on the best features to use. Possible suggestions include:

- 1) Sepal length.
- 2) Leaf features such as shape and length.
- 3) Flower diameter.
- 4) Overall plant features such as height and flower density.

Increasing the application's speed requires further development toward optimizing human interaction. Ideally, the computer would extract easy-to-identify features such as petal color and a human would need only verify its accuracy and submit more difficult-to-identify features. Since accurate identification can be made with a subset of features, the computer should guide the user towards the potentially most discriminating features.

We believe the k-nearest neighbor algorithm is the best choice for this application. It allows for the use of any combination of possible features and the training time for adding new samples is negligible. One drawback to the k-nearest neighbor algorithm is speed since all computations are delayed to classification time. With our current design, distances are calculated for the entire dataset, trivial for our small database but potentially problematic for a larger dataset. Common techniques exist to reduce the number of calculations made.

A distinguishing feature of this application is user interaction. The innate segmentation skills of a human are combined with the processing power and memory of a computer to accomplish the task of image recognition. Nevertheless, image recognition is still a difficult process. By combining the strong points of both humans and computers, we believe we have created a fast, accurate, portable, and easy-to-use application to assist in this challenging task.

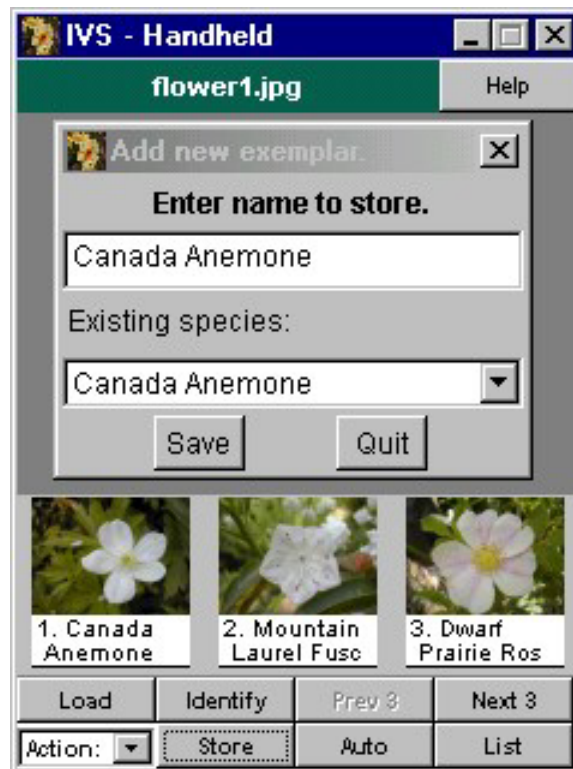


Figure 5 – Storing a new flower exemplar.

7. References

- [1] C. Enrique Ortiz, "The J2ME Mobile Information Device Profile 2.0" January 27, 2003.
- [2] G. Nagy, J. Zou, "Interactive Visual Pattern Recognition," *Proc. Int. Conf. Pattern Recognition (ICPR)*, 2002.
- [3] K. S. Fu, Y. T. Chien, and G.P. Cardillo, "A Dynamic Programming Approach to Sequential Pattern Recognition," *IEEE-EC*, 16, 313-326, 1967.
- [4] N. Norihiko, K. Hino, "An Object-Oriented Framework of Pattern Recognition Systems," Hakozaki, Fukuoka 812, Japan *Proc.*, 1988.
- [5] N. Yoshida, K. Hino, "An Object-Oriented Framework of Pattern Recognition Systems." Kyushu University, Japan, September 1988.
- [6] National Audubon Society Field Guide to North American Wildflowers. Knopf, 1995.
- [7] ORRIN E. TAULBEE and JOHN T. WELCH, JR. "A new classification theory leading to automatic pattern recognition," Goodyear Aerospace Corporation Akron, Ohio, 1966.

- [8] P. Sione, "Instance-Based Learning: A Java Implementation." October 31, 2002.
- [9] V. Piroumian "Wireless J2ME Platform Programming." Prentice Hall, October 3, 2002.
- [10] <http://www.insignia.com/content/products/pda.shtml>. Accessed September, 2002
- [11] <http://java.sun.com/j2me>. Accessed September, 2002
- [12] <http://www.sharp-usa.com>. Accessed September, 2002
- [13] <http://www.embedix.com>. Accessed September, 2002