Argus: The Digital Doorman

Rahul Sukthankar, *Just Research and Carnegie Mellon University*
Robert Stockton, *Just Research*

When you've visited someone in a large apartment or office complex, chances are that a security guard in the lobby granted you access. Perhaps over time, the guard has learned to associate you with the person you plan to visit and immediately notifies that person over the building intercom when you arrive. Argus, named after the vigilant watchman from Greek mythology, is an automated version of such a security guard: a system for automatic visitor identification. We successfully implemented and tested Argus at Just Research.

To detect visitors, Argus's digital camera photographs the building entrance at regular intervals, and a motion detection algorithm identifies potential scenes containing visitors. Using a neural-network-based face detector,[1] Argus extracts faces from these images. A memory-based face recognition system[2] examines these faces and attempts to find visually similar matches in its stored database of visitors. An interface agent notifies system users whenever visitors arrive. Users can also provide feedback to Argus in the event of misclassified visitors. Because the face recognizer can learn online, Argus immediately incorporates these corrections into its face recognition data set.

The challenge of unconstrained face recognition

Although researchers have actively studied face recognition since the mid 1960s, most systems have focused on recognizing individuals in controlled circumstances. Because reliable face recognition under unconstrained conditions is very challenging, a visitor identification system's purpose is not to control building access. For example, after Argus alerts a user that one of the user's registered guests has arrived, the user becomes responsible for admitting the guest into the building.

Figure 1 shows images captured by Argus's camera and the corresponding face images that the face detector extracted. Figure 2 shows additional images of two individuals, taken at different times of day and under different weather conditions. An individual's appearance varies dramatically with the time of day (owing to ambient lighting), weather (owing to headgear), and body pose, making automatic face recognition challenging.

Unlike other face recognition systems, Argus does not train on an initial database of labeled face images. Instead, Argus learns to identify visitors gradually as users assign names to "unknown" visitor images or register an interest in a particular visitor's arrival. Rather than attempting to recognize an individual from a single labeled image (a difficult task for both computers and humans), Argus models the variety in a person's appearance by storing several photographs for each known visitor (as many images as users care to label). Argus continues to incorporate images as users provide feedback. Our experiments have shown that this approach works well for visitor identification and that Argus can quickly improve its recognition accuracy for a person even though it acquires the images in an unconstrained setting.

How Argus works

We've implemented Argus as a distributed collection of agents linked by JGram multiagent pipelines[3] (see Figure 3). This Java-based architecture lets us distribute the system's image-processing components over several workstations and allows the easy integration of subsystems running on different operating systems.

The *delegator* agent operates as a facilitator for the agent community, providing service lookup and forwarding messages between agents as necessary. Additionally, the delegator performs some low-level image processing. The agent collects images from the camera at regular intervals and examines frames to determine whether the image is likely to contain visitors. To do this, it compares the current image with a background image that is updated slowly throughout the day. Images that pass through this filter then pass through a JGram pipeline to the *detector* and *recognizer* agents. If the recognizer matches an individual in the image with a person in the database, the delegator determines the list of users who have registered an interest in this visitor's arrival and broadcasts a message to

1094-7167/01/$10.00 © 2001 IEEE

them. The *notifier* agent then pops up a message on the user's workstation (see Figure 4). If the user does not interact with the message window, the notifier removes it after several minutes to avoid cluttering the user's desktop.

In the event of a misclassification, the user can query Argus for additional information. For instance, in Figure 5, Argus

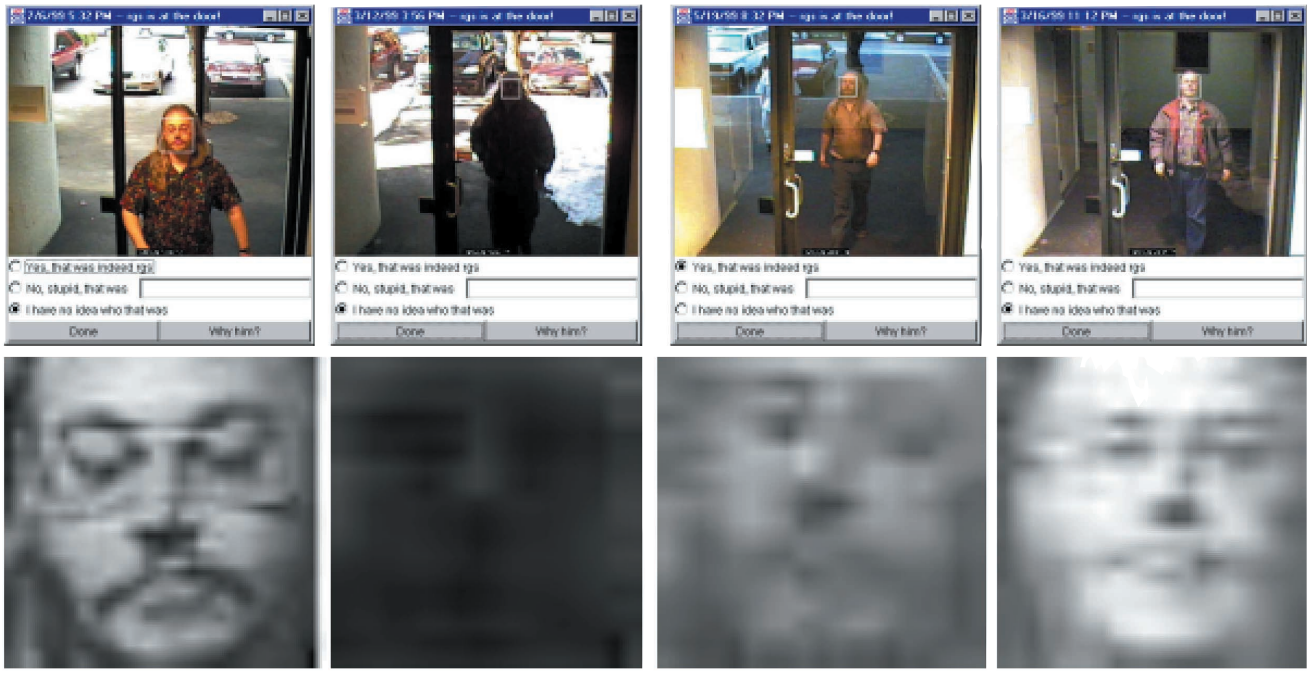

Figure 1. The top row shows photographs that Argus captured of the same person. The bottom row shows the corresponding faces that the face detector extracted, illustrating the face recognition challenges that this application poses. Note how an individual's appearance changes drastically depending on the circumstances. The face images are poor in quality because the visitor's face occupies only a small region in the original image and the images are captured in suboptimal lighting. Argus correctly identified all these images.

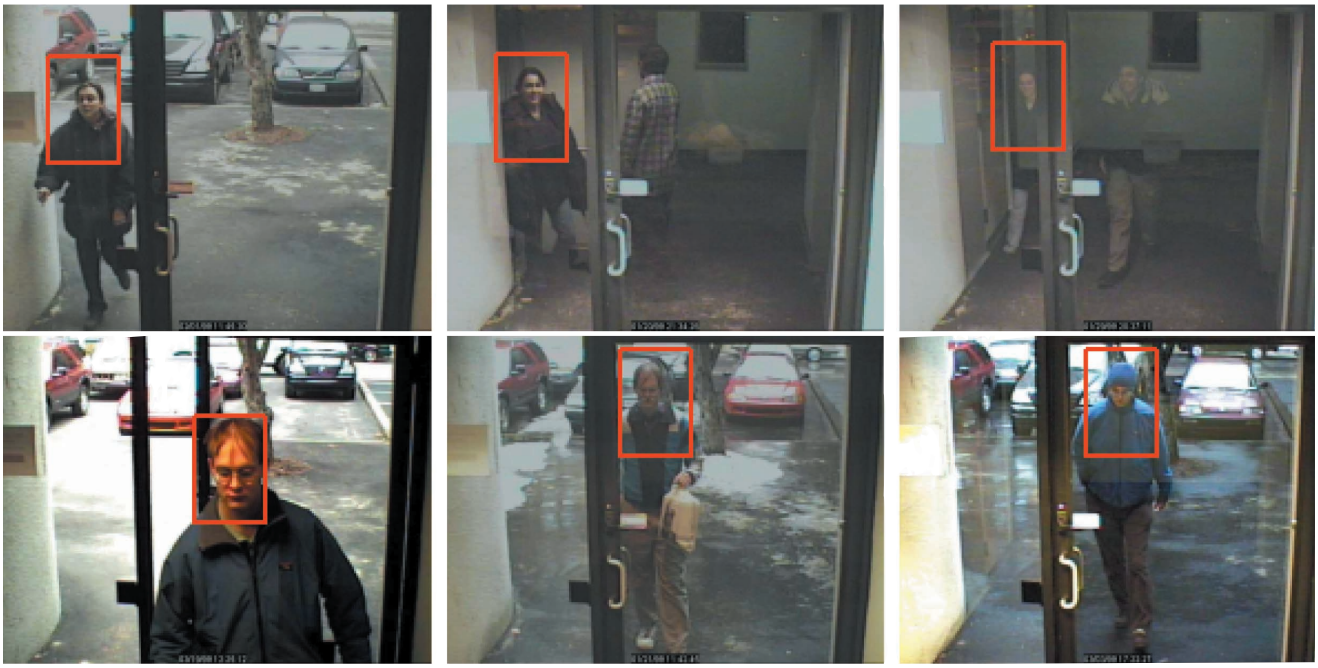

Figure 2. Additional images of two visitors. The top row illustrates how appearance changes with time of day as internal reflections from the glass door are superimposed over the individual's face. The bottom row shows appearance changes due to variations in pose and weather.

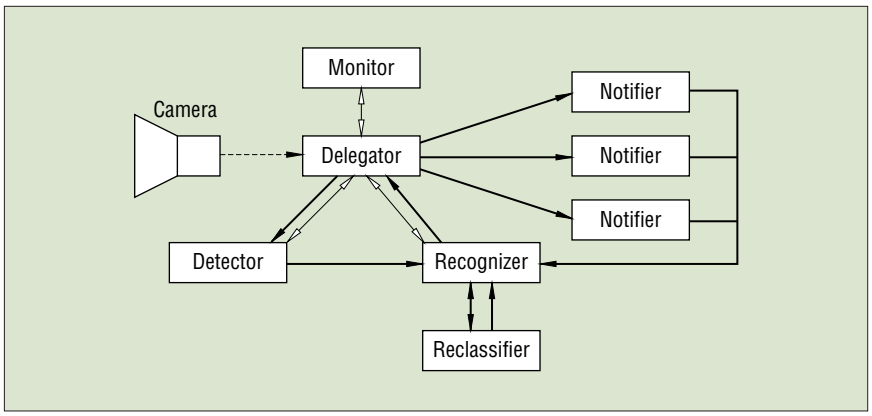

Figure 3. The Argus architecture. Each box depicts an agent. The heavier lines show the major data pathways, and the light lines show monitoring information. A line with double arrows represents a synchronous exchange, while one with a single arrow indicates asynchronous data flow.

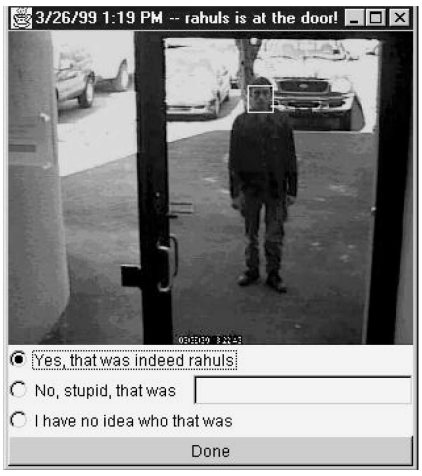

Figure 4. The Argus notifier displays an image captured by the security camera along with a box surrounding the visitor's face and a tentative identification. User feedback helps Argus improve face recognition.

Figure 5. Argus mistakenly identifies the infrequent visitor "ives" as the frequent visitor "rgs." The user asks the system, "Why him?" and Argus displays an image of the known visitor. From these photographs, you can see the visual similarity between the visitors: both wear glasses and have beards and similar hairlines. The user correctly labels the misclassified image, and Argus immediately incorporates the new image into its recognition database.

mistakenly identified the infrequent visitor "ives" as the frequent visitor "rgs." Note that, in low-quality photographs, these visitors look similar: both have beards, wear glasses, and have similar hairlines. The user can provide the correct label, and Argus will learn to distinguish between "ives" and "rgs" over time.

Administrators can periodically examine the list of unidentified visitors, using the *reclassifier* agent to provide additional training data, to correct any labeling errors. Teaching Argus about a new visitor is straightforward and requires no offline training. A user or administrator simply labels unknown or misclassified images of the new person, and Argus starts using the additional information immediately. Similarly, the *monitor* agent lets users interactively query the evaluation metrics that the other Argus agents have collected. This is particularly useful when the recognizer is simultaneously evaluating several experimental face recognition algorithms.

Face detection

The goal of face detection is to examine a camera image and extract regions containing a face. Argus employs Carnegie Mellon University's Rowley-Baluja-Kanade face detector.[1] This detector is a neural network that exhaustively scans square regions at various scales in the image to determine whether the region is a frontal, upright human face. We chose this face detector because it offers the best compromise between detection accuracy and processing speed. The limitation on face orientations (frontal and upright) is not a serious problem in this application, for two reasons. First, visitors walking toward a building entrance typically face Argus's camera. Second, the camera acquires many images of a visitor while he or she is waiting, and at least a few of them tend to be frontal. The neural network weights were trained to locate faces from Web photographs; they are not specialized for visitor identification. False negatives from the face detector (failure to detect valid faces in an image) are not a major problem for Argus because a visitor's arrival generates several images. However, the face recognizer must reject false positives (incorrectly reporting a face where none exists) to prevent incorrect visitor notifications; we discuss this further in the next section.

The face detector was trained to locate faces of any size and at any position in the image. However, Argus's camera geometry constrains the search space (visitors' faces never occupy more than a 100×100 pixel window, nor do they appear in the image's bottom quarter). So, the face detector can process a single image in approximately one second on a standard computer. While

this is too slow for real-time processing of the video stream, the face detector works adequately for Argus in combination with the motion detection filter. The detector agent passes the face detector's output (a list of regions in the image) through the JGram pipeline to the recognizer. Because the face detector locates very tightly cropped faces (typically without hair and ears), Argus enlarges the recommended region by 140 percent in each dimension before extracting the face image.

Face recognition

A memory-based face recognition algorithm works well for visitor identification for three reasons. First, because the set of visitors is not known a priori, a face recognition algorithm should be able to accommodate such changes easily. Memory-based algorithms[4] do not have an explicit training phase and are ideal for incremental training. Second, the system should acquire training data without asking the visitors to pose for photographs under controlled conditions. A memory-based system does this by collecting several images per visitor, spanning a variety of illumination conditions. Fortunately, nonparametric approaches, such as nearest-neighbor matching, typically perform well even if these images do not form a single cluster in the feature space. Finally, because the face recognition is only for notification rather than access control, the cost for misidentification is not severe. So, users might tolerate poor initial accuracy on a new visitor because recognition accuracy improves as the system receives additional feedback.

The Argus recognizer agent uses Arena, a view-based face recognition system.[2] The training phase constitutes these steps:

1. Minimize illumination variations using histogram equalization.
2. From each training image, generate a set of additional synthetic training images by making small perturbations to the original (rotation up to ± 5 degrees, scale up to ± 5 percent, and translation up to ± 2 pixels in each direction).
3. Create reduced-resolution (typically 16 × 16 or 32 × 32) versions of these images, and store them in the face database.

The addition of synthetic training images compensates for registration errors in the detector agent. Experiments have shown that this compensation produces a small but significant improvement in classification accuracy, particularly when a large selection of labeled images for a visitor are unavailable.

Figure 6. The top row shows sample images of two people that the face detector extracted. The image quality is poor owing to lighting and camera placement. Note the variation in appearance due to differences in illumination, pose, and facial expression. The bottom row shows the corresponding Arena reduced-resolution images (16 × 16 pixels).

Arena creates the low-resolution images by simply averaging over nonoverlapping square regions in the input image. This has three benefits. First, because the low-resolution image's pixels have smaller variances than do the original image's pixels, the averaging process reduces noise in the stored images. Second, the low-resolution images are less sensitive to slight variations in the original image, such as subtle facial expressions. Third, the dimensionality reduction significantly improves the performance of the nearest-neighbor algorithm, which performs poorly in very high-dimensional spaces.

The classification phase is similarly straightforward:

1. Preprocess the input image using histogram equalization.
2. Create a reduced-resolution image as we previously discussed.
3. Return the label of the single nearest neighbor to the input image, among the stored low-resolution images.

Figure 6 shows sample faces that the face detector extracted from images, along with the corresponding (enlarged) Arena reduced-resolution images. One subtle but important point is noteworthy: face recognition accuracy is sensitive to the similarity metric (or distance function) that the nearest-neighbor algorithm employs. Nearest-neighbor algorithms typically employ the L_2, or Euclidean, metric (the difference in intensity at each pixel is squared and summed over the entire image). However, the Euclidean metric severely punishes outliers. So, Arena employs the $L_{0.5}$ "distance" (the square root of the difference in intensity at each pixel is summed over the image). This distance function is more forgiving of occasional outlier pixels (such as those caused by specular reflections in the camera image).

Although the Arena algorithm is extremely simple, it outperforms established face recognition algorithms such as Principal Components Analysis (commonly known as *eigenfaces*) on standard data sets and in our visitor identification task. Extensive experiments with Arena and PCA-based methods on both Feret and ORL data sets appear elsewhere.[2] Furthermore, research has shown that Arena is computationally superior to eigenfaces and requires less storage for its database.

Argus's memory-based approach to face recognition offers two additional benefits. First, representing heterogeneous classes is straightforward. Because a single label might correspond to several visually dissimilar face images, you can associate a group of people (for example, building security personnel) with a single label (cor-

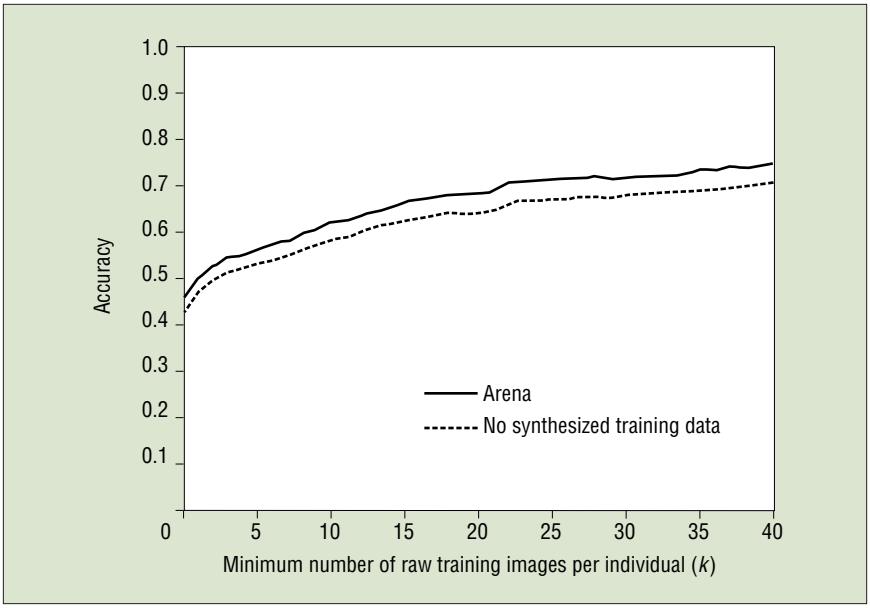

Figure 7. The variation in face recognition accuracy with respect to *k*, the minimum number of raw training images per individual in the database. The dashed line shows Arena's accuracy without synthesized training data.

responding to a single Arena class). Second, the memory-based representation can also model nonface images, such as false positives from the face detector. Argus simply labels false-positive images as "nobody," and Arena learns to reliably filter them out. In the Argus environment, false positives are typically due to a small number of events such as leaves blowing in the wind, moving cars, or patterns in the pavement generated by unusual lighting. So, Argus can treat the "nobody" class as a heterogeneous class modeled by a manageable set of exemplars.

Testing Argus

To evaluate Argus, we performed three experiments. In accordance with Feret test[5] terminology, we call the set of labeled training images the *gallery* and a test face (which must not appear in the gallery) the *probe*. These experiments used face images that Argus collected over several months.

As Figures 1 and 2 show, the photographs are very challenging, for several reasons. An outdoor camera took the images in a variety of lighting conditions, at different times during the day, and in very different weather conditions (including snow and rain). Because visitors were not instructed to pose for the camera, their head orientations varied significantly, and many of them were wearing accessories such as hats, hoods, or sunglasses. Several of the night images resulted from internal reflections of individuals as they approached the glass doors from the building's interior. The extracted face image sizes ranged from 33×33 to 100×100 pixels, with a median face image size of 47×47. Face images at this resolution are difficult to identify in the absence of other cues, even for humans.

Leave-one-out tests

The *leave-one-out* (LOO) tests used each face in the gallery as a probe image. We temporarily removed the probe and its synthesized images from the gallery and asked Argus to identify the individual in the probe. We measure accuracy as the fraction of probes that Argus correctly classified. We conducted two versions of this experiment: one with almost all the stored faces (973 images from 44 individuals) and one with only photographs of the most common visitors (881 images from 23 individuals). On the first version, Argus's overall classification accuracy was 64.1 percent (60.4 percent without synthetic images). In the second version, which focused on the task of identifying regular visitors, Argus correctly identified 69.7 percent of probes (65.2 percent without synthetic images). Argus correctly identified the two individuals with the largest number of images in the gallery (approximately 100 each) with approximately 90 percent accuracy.

Online training

The LOO tests inaccurately model visitor identification in one important respect. When multiple images of a person, all taken within a short time span, are in the gallery, a probe selected from this subset will likely be very similar to a labeled image in the gallery. However, in a real visitor identification task, all these images would have appeared together as probes, and none of the probe images would be eligible to match the other (visually similar) images in its batch. The online-training experiment addressed this potential criticism by more faithfully simulating the sequence of online training that occurs in Argus.

We initialized the gallery to be the empty set. We then successively fed the stored images (as probes) to Argus in time stamp order. During face recognition, we restricted the matching to images that had been acquired at least five minutes before the probe image's time stamp. This prevented Arena from exploiting any similarity among images taken at approximately the same time.

The online-training experiment explored how recognition accuracy varies with *k*, the minimum number of raw training images (per individual) in the database (see Figure 7). For example, when $k = 20$, Argus retains only the 11 most common visitors (each with at least 20 labeled images in the database) in the gallery. Recognition accuracy improves as *k* increases, showing that Argus is fairly accurate over a limited subset of visitors. The dashed line shows that, without synthetic images, recognition accuracy lags by three to five percent.

Robustness tests

Argus was operational at Just Research from January 1999 until March 2000. The system became quite responsive: the notifier typically displayed a window within seconds of a visitor's arrival, and occasionally before the visitor had even pressed the doorbell.

The observed recognition accuracy for common visitors (individuals with 10 or more training images) was 53.9 percent. For the full case, even including visitors for

which no training data exists, the observed accuracy was 43.4 percent.

To explore Argus' resilience to distractors, we added approximately 1,500 spurious images of faces collected by a Web spider to the database of 750 actual camera images (as a single heterogeneous class, labeled "stranger"). This change reduced the observed accuracy only by 3.1 percent. Although Argus occasionally misclassified new visitors and infrequent visitors (with few examples in the database) as strangers, the distractors had surprisingly little detrimental impact on the regular-visitor (or overall) recognition accuracy. This strengthens our hypothesis that Arena's simple view-based nearest-neighbor classification scheme is more robust than expected.

Improving Argus

Argus's recognition accuracy could be improved in two ways: using higher-quality input data and supplementing face recognition with additional visitor information.

Argus's security camera images (see Figures 4 and 6) are fairly poor in quality. The combination of small size (the visitor's face occupies only a small area in a wide field of view) and poor contrast due to uncontrolled lighting result in face images that cannot reliably be identified, even by humans familiar with the individual. So, improving camera position, installing controlled lighting, and asking visitors to pose would improve Argus' recognition accuracy. Recent experiments indicate that you can also improve face recognition performance by adding photometric variation to the synthetic training examples. Given a face image taken under one lighting condition, we can generate synthetic images showing the same individual under different lighting conditions. Because histogram equalization cannot remove the effects of lighting variation, these synthetic images could help Arena recognize visitors in difficult lighting conditions.

We have found that humans exploit additional cues to make intelligent guesses about a visitor's identity when the face image is poor. For instance, a human can take advantage of the knowledge that his spouse wears a black jacket and normally arrives around 7 pm. Argus could be extended to incorporate color models for visitor clothing and distributions of arrival times into its decision making. Argus does not exploit temporal information in the video stream; better recognition accuracy might be achieved by analyzing the visitor's motion or gait.

Argus applies a combination of image processing, machine learning, and messaging technologies to address the real-world task of visitor identification. However, Argus is still far from becoming the digital equivalent of its mythological, hundred-eyed namesake. Further research in unconstrained face recognition is needed before we can trust the digital doorman with the key to our front door. ■

Acknowledgments

Antoine Brusseau and Ray Pelletier at Just Research collaborated with us in the development of the JGram agent system. Terence Sim, Shumeet Baluja, and Matthew Mullin, also at Just Research, collaborated in the development of the Arena face recognition system. We thank Henry Rowley for his help with the CMU face detector.

References

1. H. Rowley, S. Baluja, and T. Kanade, "Neural Network-Based Face Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, Jan. 1998, pp. 23–38.
2. T. Sim et al., "Memory-Based Face Recognition for Visitor Identification," *Proc. Fourth IEEE Int'l Conf. Automatic Face and Gesture Recognition*, IEEE CS Press, Los Alamitos, Calif., 2000, pp. 214–220.
3. R. Sukthankar et al., "JGram: Rapid Development of Multi-Agent Pipelines for Real-World Tasks," *Proc. 1st Int'l Symp. Agent Systems and Applications/3rd Int'l Symp. Mobile Agents* (ASA/MA 99), IEEE CS Press, Los Alamitos, Calif., 1999, pp. 30–40.
4. C. Atkeson, W. Moore, and S. Schaal, "Locally Weighted Learning," *AI Rev.*, vol. 11, nos. 1–5, 1997, pp. 11–73.
5. P. Phillips et al., "The FERET Database and Evaluation Procedure for Face Recognition Algorithms," *Image and Vision Computing*, vol. 16, no. 5, 1998, pp. 295–306.

Rahul Sukthankar is a researcher at the Compaq Cambridge Research Lab (formerly DEC CRL) and holds a courtesy appointment as adjunct faculty (in robotics) in the School of Computer Science at Carnegie Mellon. His research focuses on computer vision and machine learning, particularly for face recognition and innovative user interfaces. He was previously a research scientist at Just Research. He received his BSE summa cum laude in computer science from Princeton and his PhD in robotics from CMU. Contact him at Compaq Research, One Cambridge Center, Cambridge, MA 02142; rahuls@cs.cmu.edu; www.cs.cmu.edu/~rahuls.

Robert Stockton is a distinguished engineer at Whizbang! Labs and was a research engineer at Just Research and Carnegie Mellon University. His areas of interest include computer vision, information extraction, and performance engineering for large machine-learning systems. He received his BA in mathematical sciences from Johns Hopkins and his MS in computer science from Carnegie Mellon. Contact him at WhizBang! Labs East, 4616 Henry St., Pittsburgh, PA 15213.