Iris Recognition for Continuous Biometric User Authentication

Author: Justin Weaver
Date: May 30th, 2011

Mentor: Kenrick Mock
UAA – Computer Science Department

Table of Contents

Note that all the images captured from the eye tracker in this document are for illustration purposes only. Consequently, they have each had their iris patterns smeared and scrambled to eliminate the possibility of any future security concerns.

Abstract

(top)

Identity management is increasingly important in today's technology-oriented world. Unfortunately, modern security techniques are far from flawless, and infallible identity management remains the "holy grail" of computer security. The aim of this project was to strengthen and enhance computer security by developing a proof-of-concept computer system to silently, unobtrusively, and continuously verify the identity of the current operator using biometric authentication in the form of iris recognition. The main novelty in the project is the design of the software, which periodically re-checks the identity of the current user. In the end, the system that was developed provides adequate proof-of-concept to legitimize further efforts in this area. However, the criteria for success laid out in the initial proposal was not completely achieved.

1. Introduction

(top)

Security, in general, is of critical concern to both private businesses and government operations alike.[R1] Stubbornly, as the information age charges forward, perfect identity management remains an intractable problem. For example, a key can be lost or stolen, and a password can be forgotten or compromised. Other singular methods of authentication suffer from similar weaknesses. At present, no single form of identification seems sufficient to unequivocally confirm the identity of a legitimate user; the next logical step is multimodal security.

Multimodal security employs more than one method of authentication to identify a user. For example, a multimodal system could require both a key and a password. However, such a system is still vulnerable, if both key and password are individually circumvented. Luckily, passwords and keys are not the only tools that security systems can employ to recognize people. Biometric authentication is ideal for use in a multimodal system.

Ideally, a computer system would automatically and reliably recognize the user, without requiring their cooperation, much the same way another human being would. Loosely stated, biometric authentication is the idea that a machine can use input devices to measure a person's physical characteristics and/or behaviors to accurately identify them.[R2] The data that a biometric authentication system uses to identify a user is difficult to lose, copy, fake, or steal.

Most modern authentication systems also demonstrate an additional weakness one could refer to as the "gatekeeper" problem: once a user has been authenticated, they are assumed legitimate for the remainder of their interaction with the system. Unfortunately, such a model promises smooth sailing to any attacker, once they are past the initial login. It also leaves the system in a state vulnerable to a user switch. The most brutish solution would be to simply prompt the user to re-authenticate periodically. However, this is less than ideal, since it would pester the user with unwelcome and inconvenient interruptions.

However, using biometric recognition, a system could periodically (continuously) re-verify the user's identity in a passive manner. Passive, meaning that the system would not require the user's cooperation, or even their awareness, to authenticate them. The notion of continuous authentication can be employed to significantly reinforce existing security techniques.[R3][R4][R5] If the operator is not the person the system expects them to be, or if the operator disappears, the machine can take action (e.g., lock itself, send a notification, or trigger an alarm).

Iris recognition is an ideal biometric, because it can be implemented in a passive way, and it is

incredibly accurate.

Because iris patterns are remarkably unique,[Illustration 1] the techniques of Dr. John Daugman[B1] allow stored iris images to be matched with astonishing accuracy against the images from a live high resolution video camera.[R6]

A study of Dr. John Daugman's algorithms successfully completed 200 billion cross-comparisons without a single false match.[R7] The database for the study came from the United Arab Emirates (UAE), where they have been using Daugman's iris recognition algorithms to identify people crossing their border since 2001. The database contained 632,500 unique samples, and spanned 152 nationalities.

Illustration 1: Patterns in the Human Iris. Picture taken by Sarah Cartwright, 2007.

Iris recognition is better suited to continuous biometric authentication than other methods, not only because of its accuracy, but because the required equipment is unobtrusive, and the data used for identification of the user is constantly available. Methods like keystroke analysis, for example, can be employed to match the current user's typing pattern to a stored profile, but such techniques are only marginally accurate at best; additionally, the use of a computer doesn't necessitate the constant use of a keyboard, so the data required to identify the current user is not always available; which is a requirement for passive recognition.[R8][R9] Fingerprint recognition, likewise, suffers from the same defect, because it requires that the user's hand be in nearly constant contact with a measurement device. However, passive iris recognition holds an advantage: when using the terminal, the user will undoubtedly spend most of their time looking at the screen. So, the system is designed to expect many failed matches (note: not mismatches), but requires at least one good match per arbitrary time period.

In summary, a strong multimodal system could require an initial password and then proceed with continuous iris recognition using a remote eye tracker. Such a system could be extremely useful for highly sensitive workstations.

2. Overview

(top)

The goal of this project was to develop a proof-of-concept system that periodically, passively verifies the user's identity using a remote eye tracker, the established iris matching algorithms of Dr. John Daugman, and the prior work of Dr. Kenrick Mock[B2] and Dr. Bogdan Hoanca.

2.1 Remote Eye Trackers for Iris Recognition

(top)

Normally, eye trackers are used as assistive technology, i.e., to facilitate computer access for the disabled. However, the specifications of these devices often make them appropriate for iris recognition.

Illustration 2: An EyeTech TM3 eye tracker. Picture from the EyeTech website.

A remote eye tracker, like EyeTech's TM3, (as opposed to an intrusive head-mounted one) is essentially just a high resolution near-infrared (700-900nm wavelength) light video camera situated between a pair of near-infrared spotlights.[R10][Illustration 2]

The eye tracker is placed in front of the user in an unobtrusive manner, usually just below the computer monitor.[Illustration 3] The near-infrared light is completely invisible to the human eye, but the eye tracker's special camera is able to see it clearly. In near infrared lighting conditions, rich patterns are visible in the iris.[R11]

Illustration 3: EyeTech TM3 eye tracker mounted on LCD monitor. Picture from the EyeTech website.

Once activated, the eye tracker switches its spotlights on, which cause a pair of "glint" points to appear in the video image of the user's eyes. [Illustration 4] The eye tracker then uses those glint points to orient itself and determine where the user is looking at any given time.

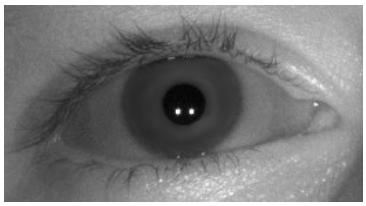

Illustration 4: The glint points in the user's eyes from the infrared spotlights as seen by the eye tracker's camera.

Through its driver software, the eye tracker provides a raw image of the user's face,[Illustration 5] the coordinates of the user's pupils, and the previously mentioned glint-points. This means that sample images need only be examined when the eye tracker finds an eye. It also means we can effortlessly minimize the area of interest within the image to a very small region immediately around the provided pupil coordinates. Furthermore, a certain degree of image focus is guaranteed by the mere fact that the eye tracker was able to locate the user's eye.

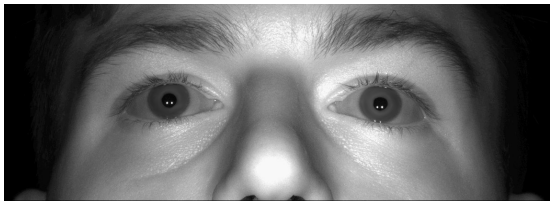

Illustration 5: A raw image returned from the eye tracker.

The fact that the camera is sensitive only to the near-infrared spectrum light helps to reduce the potential for lighting variations. Commonly used incandescent light bulbs, do not produce light in the near-infrared spectrum. Additionally, florescent bulbs produce it only in marginal quantities. Thus, without its spotlights, the eye tracker's camera would often see only blackness. The most noteworthy exception to the rule is Sunlight, which does, in fact, contain intense quantities of the spectrum in question, and thus should be kept to a minimum. Even if the lighting variation does not effect the actual matching, it may preclude it by causing the eye tracker to fail when it attempts to find the glint points in the user's eyes.

To summarize, the EyeTech TM3 eye tracker's high resolution near-infrared camera, two near-infrared spectrum spotlights, and unobtrusive nature, make it ideal for the purpose of passive iris recognition. Additionally, the eye tracker provides some data that makes the process of recognizing eye features easier.

3. Requirements

(top)

3.1 Base Requirements

(top)

- An EyeTech TM3 eye tracker.
- A personal computer with at least these minimum requirements: Pentium 800Mhz, 1 available IEEE 1394 port, 128MB RAM, 50MB free disk space, Windows XP or Vista. [R12]

3.2 Software Requirements

(top)

1. The software must run on the Microsoft .NET framework.[B3]
    - Working with Dr. Kenrick Mock, I have previously developed a software wrapper that provides a way to control EyeTech's line of eye trackers from within the .NET environment. I named the wrapper QuickLinkDotNet, and released it under an open source license.[B4] So, this project continues and extends that work.
2. The software should use the iris matching algorithms of Dr. John Daugman.
3. The software cannot place an obnoxiously heavy burden on the system as a result of the periodic iris checks, i.e., the system's user interface will not become noticeably slower or cumbersome to operate.

3.2.1 Software Behavior Specifications

(top)

The software runs in a normal Windows user session. It displays a small, draggable readout window, and then begins to silently monitor input from the eye tracker. Periodically, the program checks for the presence of a user sitting in front of the computer. When a user is detected, the readout displays the identity of that user, if that user's profile is stored on the system; otherwise, it reports that the current user is unknown. The known users consist of a small database containing samples from the student researcher and his mentors.

By accessing the eye tracker's API through the wrapper, the program uses the location of the user's eyes within the raw image to easily locate their eyes. The software then proceeds to extract and compare the user's irises against each user profile stored in the database, one at a time.

3.3 Other Requirements

(top)

The system must be minimally intrusive to the user, i.e., the user does not have to wear any special equipment, sit in an uncomfortable position, nor behave in an overly awkward manner to use the system.

3.3.1 Demonstration Requirements

(top)

The success of the project was originally specified by the reproducible demonstration of the steps listed below.[B5] The planned demonstration involved running the project's software in the background of a normal Windows session.

3.3.1.1 Demonstration Steps

(top)

These steps require the participation of three people: two “known” users, and one “unknown” user. In practice, the three users will most likely consist of the student researcher and the project mentors.

1. Begin with student researcher sitting in front of computer. The system is accurately and continuously identifying the student researcher.
2. The student researcher looks away for 1-4 seconds, after which he looks at the monitor again. The system continues to recognize the student researcher even though he looked away, because the system allows a grace period. If he looks away for more than 5 seconds, then the system reports that there is no user. Throughout the subsequent steps, the 5 second grace period is always implied.
3. The student researcher stands up. The system accurately recognizes there is no current user.
4. The known user sits down. The system accurately identifies the known user.
5. The known user stands up. The system accurately recognizes there is no current user.
6. The unknown user sits down. The system detects and reports the unknown user.
7. The unknown user stands up. The system accurately recognizes there is no current user.
8. The student researcher sits down. The system accurately identifies the student researcher again.

4. Methodology

(top)

Note that, for added speed and simplicity, I decided to perform matching on only the user's right eye. Additionally, for this proof-of-concept system, no attempt was made at even the most rudimentary database optimization. The project focused on a very small set of samples, so database optimization was not a high priority. Instead, a simple linear search was used.

4.1 Iris Matching Steps

(top)

1. Find and extract the pixels that make up the portion of the iris we are interested in; this includes constructing a mask to hide noise, i.e., anything that obscures or compromises part of area of interest within the iris.
2. Normalize the iris image to account for size variations caused by camera distance, pupil dilation, etc.
3. Extract data from the normalized iris by taking patches of the image in a regular pattern and performing a special mathematical transform on them to build a hash table.
4. Perform the matching calculation by finding the sum of the differences in hashes.

4.1.1 Step One: Iris Image Extraction

(top)

The method of iris extraction involves extracting only the "zig-zag" collarette region of the iris, immediately adjacent to the pupil.[Illustration 6]

The collarette contains the most distinct portion of the iris pattern, which makes a high degree of matching accuracy possible using only this small portion of the overall iris.[R13]

The width of the collarette region does not vary with pupil dilation, and it's always centered exactly around the pupil; which the iris as a whole often is not.[R6][R7]

While eyelids and eyelashes can still obscure portions of the collarette, the most common source of noise occurs when the user's eye is oriented in such a way that the glint points from the eye tracker's spotlights fall onto the collarette region. To account for noise, a special mask is generated to track which pixels should be ignored. However, the only noise currently taken into account is the aforementioned glint from the eye tracker's spotlights. Other sources of noise (e.g., eyelids, eyelashes) are currently not detected, and thus not filtered.

To further simplify image analysis, I used a free open source library, called AForge,[B6] which provides utilities for computer vision and artificial intelligence. To locate the pupil within the cutout image, I used AForge's threshold "blob" detection[B7] (i.e., the largest, darkest blob). Then, I simply transform the blob into a best-fit circle. I also used blob detection to define the glint points.

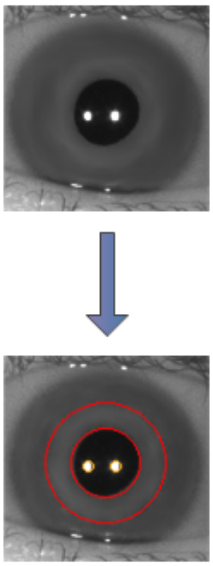

Illustration 6: An eye with the extracted region and glint points highlighted.

4.1.2 Step Two: Iris Image Normalization

(top)

Normalization helps to account for variations caused by the distance between the eye tracker and the user's eyes – as well as variations in their pupil dilation. The normalization process is also referred to by Daugman as the "homogeneous rubber sheet model," because it stretches the iris image onto a statically sized rectangular area,[Illustration 7] by interpreting the isolated iris pixels according to polar coordinates centered on the pupil.[R6][R7]

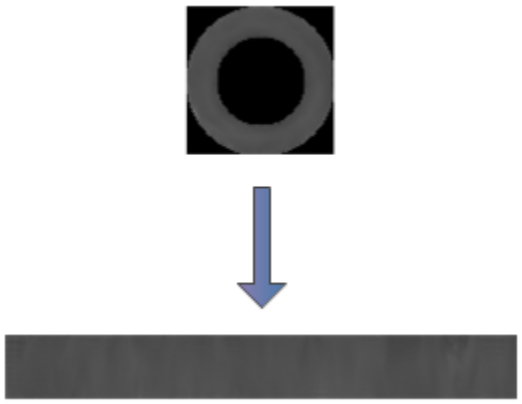

Illustration 7: Normalization of an extracted iris image.

If a mask was generated for the iris image in question (i.e., the glint points appeared in the collarette region of the iris – see 4.1.1 Step One: Iris Image Extraction), then the mask must be translated along with the iris image itself. The mask is used later during the hash generation and matching steps.

To perform the normalization, a radial "slice" of the original image is taken, starting from the center of the pupil, for each column in the statically sized output image. Then each slice is further subdivided into even smaller "tiny" slices. Each of those tiny slices is examined, and the color values of the pixels that they touch are noted in the form of lists. Then the collection of lists from those tiny slices have their lengths normalized to the height of the output image via simple interpolation. Finally, the collections of tiny slices are averaged together, slice-wise (as one might zip up a jacket). The resulting list of averaged values is written into the corresponding column of the output image, and the algorithm proceeds to the next slice.

4.1.3 Step Three: Hash Generation

(top)

To summarize its function very generally: the algorithm takes the normalized image, and runs Gabor wavelet transforms over little square pieces of it, in many overlapping locations of various size, to extract what amounts to an iris "bar-code." Any masked areas of the normalized iris image are taken into account during this step. Unfortunately, this step is only partially finished (see 5.2 Inherited Code, Hash Generation, and Angle Invariance).

4.1.4 Step Four: Hash Matching

(top)

The matching is done by calculating the Hamming distance between two iris hashes. To calculate the Hamming distance, the two hashes are compared value-by-value.

The Hamming distance is calculated as the ratio of the number of mismatched values over the number of significant (i.e., unmasked) values, and reported as a number between 0 and 1, as described by Daugman.[R6][R7]

Finally, we decide if two hashes match based on whether the Hamming distance is below a certain threshold value.

5. Results

(top)

The system is incomplete, so the software does not implement the exact behavior described in the proposal. Consequently, the demo and other planned qualitative evaluations cannot currently be performed. However, the software still provides legitimate proof-of-concept, because it establishes the feasibility of continuous iris recognition using a remote eye tracker.

With our tiny database of four users, the software consistently recognizes me and differentiates me from the other users during live, and prerecorded, test sessions when the Hamming distance threshold is set to 0.25.[Illustration 8] However, the software does not achieve angle invariance (see 5.2 Inherited Code, Hash Generation, and Angle Invariance).

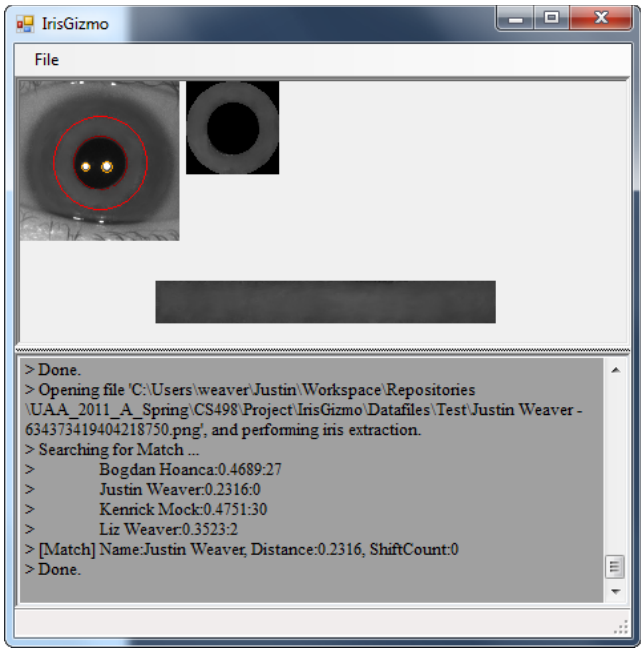

Illustration 8: A screenshot of IrisGizmo in action.

Obviously, the system needs more work. To that end, I designed the software carefully to leave behind a solid, well-documented foundation from which to continue development.

5.1 Deviations from Planned Software Behavior

(top)

Most of the deviations in planned software behavior are due to the developmental nature of the program at this stage. A list of deviations is provided below.

1. *Deviation:* the software does not speed up matching (as described in the proposal) by starting the search with the last matched user; nor does it stop the search when a match is found.
    - *Reason:* I need the full, exhaustive search (while debugging) in order to verify there are no cases where more than one good match occurs.
2. *Deviation:* the software does not use a 5 second timeout period before changing its readout.
    - *Reason:* the software shows full developmental/debugging data in real-time.
3. *Deviation:* the software does not ever report the "No User" condition; instead, it simply reports "Match" or "No Match" each time it performs a check.
    - *Reason:* recognition of the "No User" condition requires the use of a timeout (see item 2: above).

5.2 Inherited Code, Hash Generation, and Angle Invariance

(top)

I inherited three algorithms from a previous UAA student group. The three algorithms were: the iris normalization, the Gabor hash computation, and the Hamming distance calculation. Their work was done in the Java programming language, so I hand-ported their code to C#.

We expected their code to save a great deal of time and effort by providing some of the more complex portions of the project's software. Unfortunately, I eventually discovered that their code was designed only to work with a very small set of samples, and therefore bore only a superficial similarity to Daugman's algorithms. Consequently, their code was unsuitable for our needs, and had to be rewritten from scratch. I rewrote the algorithm to normalize iris images, and the algorithms to perform Hamming distance calculations, but I did not have time for a detailed implementation of Daugman's hash generation algorithm.

I decided to simplify matters by creating a "stub" (a temporary patch) hash generation method using parts of the previous groups code, along with a touch of my own ingenuity. The stub code does a primitive, pixel-by-pixel comparison of filtered, normalized iris images -- in contrast to Daugman's superior bar-code method. The advantage the of the stub is its tolerance for unmasked noise, i.e., it prevents false matches (in our limited testing); the most notable disadvantage is its lack of angle invariance. In other words, it is unable to make a match unless the user's eye is oriented in much the same way it was when the database sample (i.e., the user's authentic iris sample) was recorded. Thus, the software can go long periods of time without a match, depending on the user's posture at any given time.

As part of my stub algorithm, I attempt to account for iris rotation (due primarily to the user's head-tilt) by shifting the hashes column-wise (with wrap) in each direction, and recalculating the Hamming distance; this process is repeated at incremental shifts to account for a configurable maximum degree of rotation. The shifting increased the accuracy of the Hamming distance calculation, and in some cases (observed in stored samples), even produced a match when none was achieved without the shifting.

To achieve angle invariance, Daugman's hash generation algorithm must be implemented correctly. However, if Daugman's hash generation method is to be used, then exact iris

extraction is critical.[R6] So, more noise must be detected and masked, and a more exact fit should be determined for the pupillary and collarette boundaries (see 5.4 Future Work). Furthermore, once Daugman's Gabor hash generation algorithm is correctly implemented, the current circular hash shifting will be unnecessary.

5.3 Revising an Idea from the Proposal

(top)

The original proposal mentioned, as a possible future direction of research, the potential to tune a system to take advantage of the knowledge that a user is present or absent, by using that information to decide when it would be appropriate to run resource-intensive tasks.

This is a sound idea in theory. However, this particular eye tracker offloads the work of processing new images to the computer. Consequently, the eye tracker itself has a high software overhead to begin with; and it hammers the CPU relentlessly when it cannot find a person's eyes (as it performs an exhaustive search).

On the other hand, the software could disable image processing and lock the screen when it recognizes the absence of the user for a set time period. When the user returned they would have to press a key, or otherwise notify the software. The software would then re-enable image processing, and verify the user's identity via iris recognition, before unlocking the screen. However, this is obviously less than ideal, and may necessitate short delays while the image processing is disabled and enabled.

So, given the specific eye tracker in question, the idea would likely not be a practical direction for future efforts.

5.4 Future Work

(top)

1. Implement Daugman's Gabor hash generation to achieve angle-invariant matching (see 5.2 Inherited Code, Hash Generation, and Angle Invariance). In order to accomplish this, a more precise fit for the boundaries of the iris must be achieved. Additionally, unwanted noise must be reliably detected and masked.
    - The project's software defines the pupillary boundary using a simple circle. However, this is not sufficient to truly describe a pupil, since they are usually not perfectly circular – even when the user is looking directly at the camera. Thus, slivers of valuable iris information are often trimmed out, and unwanted noise is often included. A superior algorithm would find a more exact fit for the pupillary boundary.
    - For simplicity, I assume that the collarette is a static number of pixels of width (as specified in the iris extractor's configuration file). It is true that the collarette's width does not change with the user's pupil dilation; however, its width can vary from user to user. Furthermore, as with the pupil, I assume the collarette's boundary is a perfect circle; which may not be true, especially if the user's eye is tilted at a severe angle to the camera. So, ideally one would somehow detect the exact collarette boundary. One possible method would be to employ some form of edge detection to find the collarette boundary – given that it begins at the pupillary boundary, and ends where the density of ridges in the iris declines abruptly.

- Finding and masking various other types of noise within the iris image: eyelids, eyelashes, smudges and fingerprints on glasses, etc.
- The eye tracker requires a certain degree of focus in order to detect and locate the user's eye. However, images need to meet a minimum focus criteria[R6] that may be more restrictive than that which is needed for the eye tracker to function; more testing is required to determine whether this is true. As of right now, no additional focus evaluation is performed.
- Finally, the algorithm to generate the Gabor based hash needs to be properly implemented as indicated by Daugman.[R6][R7][R11]

2. The algorithms that normalize the iris image, calculate the Hamming distance, and construct and search the user database must each be optimized for speed and storage overhead.
3. Once the software is actually finished, and the demonstration can be completed, the most immediate next-step would be a formal user study, with more human subjects, to affirm the accuracy and usability of the design.
4. To take the system beyond the proof-of-concept stage, future work could be done to actually integrate the software into the Windows login, so that the system will automatically take appropriate action when a different user (or no user) sits in front of it (lock the terminal, switch to new user, etc).
5. It may be possible to integrate iris recognition with the normal functionality of the eye tracker, to defeat any possibility of tricking the system by fixing a falsified iris image in front of the camera.
6. Ultimately, the goal is the integration of passive, continuous iris recognition with other authentication schemes to create robust multimodal systems.

6. Summary

(top)

The story of this project is largely one of success, because my efforts produced a viable proof-of-concept system. Yet I failed to meet all of the requirements I set in my proposal. Most of the deviations from planned software behavior are due to the developmental nature of the program at this (unfinished) stage.

Dr. Daugman's Gabor hash generation algorithms needs to be implemented correctly; which also requires more exact extraction of iris pixels, and more robust noise detection. Additionally, some of the other key algorithms in the system (e.g., iris normalization, Hamming distance calculation, database searching) need optimization.

The framework I have built, and the challenges I have overcome, should lend drive and focus to future research efforts in this area.

7. References and Supporting Literature

(top)

- [R1] Defense Science Board (DSB). "Report of the Defense Science Board Task Force on Defense Biometrics." Office of the Under Secretary of Defense For Acquisition, Technology, and Logistics

(Washington, D.C. 20301-3140, March 2007). <http://www.acq.osd.mil/dsb/reports/ADA465930.pdf>.

- [R2] NTSC Subcommittee on Biometrics. "Biometrics Overview." Web (August 2006). <http://www.biometrics.gov/Documents/BioOverview.pdf>.
- [R3] Chandra, A. and Calderon, T. "Challenges and Constraints to the Diffusion of Biometrics in Information Systems." Commun. ACM 48, 12 (Dec. 2005), 101-106.
- [R4] K. Niinuma and A. K. Jain. "Continuous User Authentication Using Temporal Information." Prof of SPIE, Biometric Technology for Human Identification VII (April 2010). <http://biometrics.cse.msu.edu/Publications/Face/NiinumaJain_ContinuousAuth_SPIE10.pdf>.
- [R5] Kinnunen, T., Sedlak, F., and Bednarik, R. "Towards Task-Independent Person Authentication Using Eye Movement Signals." Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications (Austin, Texas, March 22 - 24, 2010). ETRA '10. ACM, New York, NY, 187-190.
- [R6] Daugman, John. "How Iris Recognition Works." IEEE Transactions on Circuits and Systems for Video Technology 14(1) (January 2004):21-30. <http://www.cl.cam.ac.uk/users/jgd1000/irisrecog.pdf>.
- [R7] Daugman, John. "Probing the Uniqueness and Randomness of IrisCodes: Results from 200 Billion Iris Pair Comparisons." Technical Report UCAM-CL-TR-635 (University of Cambridge Computer Laboratory, June 2005). <http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-635.pdf>.
- [R8] Monrose, F. and Rubin, A. "Authentication Via Keystroke Dynamics." Proceedings of the 4th ACM Conference on Computer and Communications Security (Zurich, Switzerland, April 01 - 04, 1997). CCS '97. ACM, New York, NY, 48-56.
- [R9] Chudá, D. and Ďurfina, M. "Multifactor Authentication Based on Keystroke Dynamics." Proceedings of the international Conference on Computer Systems and Technologies and Workshop For PhD Students in Computing (Ruse, Bulgaria, June 18 - 19, 2009). B. Rachev and A. Smrikarov, Eds. CompSysTech '09, vol. 433. ACM, New York, NY, 1-6.
- [R10] EyeTech Digital Systems. "EyeTech TM3 Details." Web. Accessed 09 Dec 2010. <http://www.eyetechds.com/research/tm3-qc>.
- [R11] Daugman, John. "Importance of Being Random, The - Statistical Principles of Iris Recognition." Technical Report CB3 0FD, UK (University of Cambridge Computer Laboratory, Dec 2001). <http://www.cl.cam.ac.uk/users/jgd1000/patrec.pdf>
- [R12] EyeTech Digital Systems. "TM3 Hardware Installation Manual." EyeTech Digital Systems Website (October 2007). <http://www.eyetechds.com/assistivetech/support/downloads/TM3HardwareInstallManual.pdf>.
- [R13] XiaoFu He and PengFei Shi. "An Efficient Iris Segmentation Method for Recognition." Pattern Recognition and Image Analysis: Third International Conference on Advances in Pattern Recognition, ICAPR 2005 Bath, UK, August, 2005 Proceedings, Part II. LNCS3687. ISBN-10 3-540-28833-3 Springer Berlin Heidelberg New York

7.1 Other Resources

(top)

- [B1] Dr. John Daugman's Homepage. Web (Accessed Oct 19, 2011). <http://www.cl.cam.ac.uk/~jgd1000/>.
- [B2] Dr. Kenrick Mock's Homepage. Web (Accessed Oct 24, 2011). <http://www.math.uaa.alaska.edu/~afkjm/>.

- [B3] Microsoft Corporation. "Microsoft .NET Framework." Web (Accessed Apr 28, 2011). <http://www.microsoft.com/net/>.
- [B4] Justin Weaver, *QuickLinkAPI4NET*. Software source code: an API wrapper for Windows .NET. Web (Accessed Oct 18, 2011). <http://quicklinkapi4net.googlecode.com>.
- [B5] Justin Weaver. "Iris Recognition for Continuous Biometric User Authentication" (i.e., the original proposal for this research). Web (Accessed Apr 28, 2011). <http://www.uaa.alaska.edu/ours/success/competitive-grants/research/upload/Justin_Weaver.pdf>.
- [B6] AForge.NET. "AForge: An open source C# framework designed for developers and researchers in the fields of Computer Vision and Artificial Intelligence." Web (Accessed Apr 27, 2011). <http://www.aforgenet.com/framework/>.
- [B7] AForge.NET. AForge: Blob Processing." Web (Accessed Apr 27, 2011). <http://www.aforgenet.com/framework/features/blobs_processing.html>.