Dynamic Scalable Distributed Face Recognition System Security Framework

Konrad Rzeszutek
University of New Orleans
Computer Science Department
2000 Lakeshore Drive
New Orleans, LA 70148
krzeszut@cs.uno.edu

May 26, 2002

Abstract

Many of our daily interactions depend on being able to recognize one's face. Modeling human recognition can be used in many fields: surveillance systems, security systems, autonomous navigation of vehicles and many more.

Current face recognition technologies are extremely computationally intensive. To accommodate the load in near-real time, a multi component scalable distributed framework is presented. This framework allows separating the functionality of recognition, notification, and replay of camera-feed in different independent components. Furthermore the framework was designed with RAS (reliability, availability and serviceability), adaptability to network changes and easy path of adding (or replacing) of components in mind. Also becuase of its distributed nature, it can work over a large network (Internet).

This paper describes in detail the framework of such system.

1 Introduction

Face recognition is a paramount functionality in a human's life. Many of our daily interactions depend on being able to recognize one's face. Humans can recognize faces even when the image is occluded (a person wearing sunglasses) or distorted (as in example of a caricature). This recognition task is performed effortlessly and in a split of a second. Understanding how humans recognize is a mind-boggling question for physicians, neural scientists, and computer scientists. Many researches think that the human brain has a region dedicated to face recognition solely. The attempt to model the brain's functionality by scientists has resulted in many face-recognition techniques. Unfortunately so far the techniques are extremely slow and give erroneous results compared to human's native recognition. Recognizing a face in any environment using the latest face recognition technology does not have the same chance of success as a human has. However, in a controlled environment the success rate increases.

It might be baffling to anybody why modeling or even understanding human face-recognition is so important. Besides the cognitive aspects, understanding the under laying mechanism could be used to build a computer system capable of recognizing not only human faces but also various other items. Such system could be used in surveillance systems, biomedical field (replacing a human faulty eye), military (recognizing terrorists by special forces), warehousing (robots capable of moving around without special markings on the floor), autonomous navigation of ve-

hicles, system security (as a replacement for password authentication) and many more.

1.1 System security

The system security is of particular interest. The current approaches include user authentication through either unique physical or behavioral attributes (such as speech recognition, iris checking, thermal scanning and signature checking) or on keys, PINs or passwords. However they are all quite obtrusive.

But face recognition offers another approach - the least obtrusive of them. It's capable of validating the user without the participation of the user - therefore eliminating some form of privacy invasion. But that is not the only field in which face recognition can play an important role.

1.2 Surveillance Systems

The most interesting of these uses is the surveillance systems. After the September 11th 2001 attack on United States by Al Qaeda (the terrorist network), being able to recognize terrorists and alarm the authorities became an urgent need. Many agencies in United States believe that the solution is in tracking its citizens and making sure that people entering United States carry no weapons.

Another solution is to harness face recognition technology. The human face is the most widely used form of authentication around the world - be it on passports, driver's license, school identification cards, and many more documents. Therefore it offers the most convenient and easiest form of finding and recognizing individuals.

Furthermore, the set of faces that we wish to recognize (terrorists, criminals, etc) is quite small compared to the world population. With current technology processing a set of five thousand faces or even more does not present a problem. Though with a set in order of millions (one face can also have variations - sunglasses, beard, scars, etc) there is a performance problem. But a computer framework capable of load balancing, distributed recognition of faces and off-site storage for replay of camera feed would be the most flexible and scalable solution. Furthermore with the advancement of face recognition technologies a module system is preferred - allowing for example the replacement of the face recognition module without changing the rest of the framework.

1.3 Aim of this work

Face recognition is a very challenging problem. Images that appear like humans can be very difficult to match under different environment changes (lighting, background, clothing, etc). Furthermore, recognizing faces that might have altered (shaved off beard, sunglasses, chin and nose operation) requires expanding the basic image with more possible alternations. Therefore from one image of a terrorist, five thousand can be easily generated. With so many images, easily approaching the number of millions, a feasible framework capable of processing the images almost in real-time is required.

It is also worth noting that to date, no work using a distributed system in face recognition technique has been found and this work investigates the aspects of such system. The work shows how such system is modeled, and its workings.

2 Dynamic Scalable Distributed Face Recognition Security System Framework

The current advances in computer technology has resulted in manufacturing inexpensive, high-capacity, high-computationally-feasible systems. A cluster of these systems is capable of outpacing some of the high-performance supercomputers at a fraction of cost. Face recognition, belonging to the family of computationally intensive applications, requires high-performance systems to perform its calculations. To solve the calculations in near real-time would require expensive high-performance supercomputers or clustering inexpensive systems.

Therefore a scalable distributed multi-component framework system called Apollo is proposed. This system would augment the computational intensity requirments of face recognition matching on a large

scale. Furthermore Apollo was designed with reliability, availability, and serviceability (RAS) in mind. The reliability lies in its dynamicity and adaptability traits in changing network environment. The off-site storage and redundancy fulfill the availability profile and the serviceability lies in its scalable feature and option in replacing (or upgrading) components.

2.1 Definition of terms

- *Pool* - The collection of machines in one of the components.
- *Systems* - A collection of computer systems.
- *Services* - A set of computer services (such as web server, lookup server).
- *Clients* - Clients using the services (web browser).
- *RTP* - From RTP specification (RFC 1889) [?]: "RTP provides end-to-end network transport functions suitable for applications transmitting real-time data, such as audio, video or simulated data, over multicast or unicast network services."
- *Off-line storage* - Storing of camera-feed on a redudant system for latter replay if neccesary.

2.2 Components of Apollo

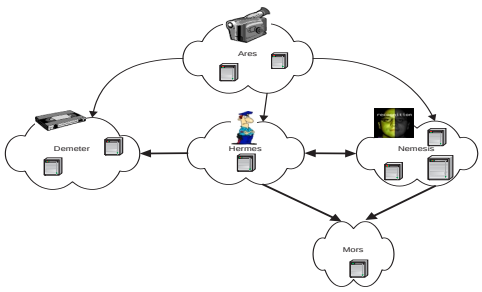

Figure 1: Components of Apollo

There are five components, which are:

- *Hermes* - the load balancer acting as *police officer* directing traffic to the appropriate components capable of handling the required requests and data.
- *Ares* - the thin client sending its camera feed to off-site storage and face recognition (Demeter and Nemesis respectively).
- *Demeter* - the off-site storage component. Stores the camera feed for future replay.
- *Nemesis* - the number crunching component - processes the images in an attempt to recognize a face.
- *Mors* - the logging / notifying component saving the trigger events (a face is recognized).

2.3 Hermes

Hermes - the messenger of Gods, though in this scenario plays a role of the "police officer" directing traffic (see Fig: 1). Hermes acts as dynamic centralized information portal. Since the system is distributed many systems can exist in a Hermes pool and provide information to its clients. The information is queried directly from the other three components: Demeter, Nemesis and Mors at every predefined amount of time. This allows for near real-time acquisition of load information, number of clients using the component, the maximum number of connections allowed and of course the address of the component.

To have such a highly dynamic system, capable of detecting changes in such dynamic computing environment, Hermes must be quite adaptive. Furthermore, Hermes by itself must be scalable too (its part of a distributed system) - therefore there can be many systems in a Hermes component pool. Each of them has the same information about the three other components, though each retrieves the information by itself.

The main purpose of Hermes is to provide information for two components: Ares and Nemesis. Ares requires information about Nemesis and Demeter, while Nemesis requires information about Mors (see Fig: 5). Hermes provides this information - giving the

client the least loaded requested component, therefore eliminating bottlenecks and keeping a fairly level load across the systems. When the load on the specific pool of components if fairly high, the solution is to add a system of that component, and the load balance will direct new traffic to that system.

In summary Hermes' solely task is to provide the addresses of the desired components with the least load. Each pool can also perform these tasks indivudually, thus in sense integrating Hermes functionality in their own component. This in result eliminates Hermes, but provides fail-safe redudancy and autonomicity.

2.4 Ares

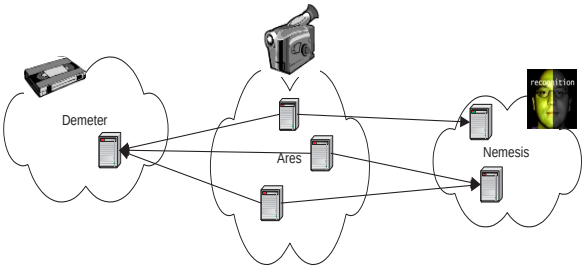

Figure 2: Ares interacting with Demeter and Nemesis

Ares - the God of War - is responsible for capturing the data feed from the camera and sending it to an off-site storage and face recognition component (Demeter and Nemesis respectively) (see Fig: 2). Since the system is dynamic, it has to be capable of finding the required components dynamically. To do so, it locates Hermes - the load balancer - and queries for the desired component. Upon receiving the addresses, it registers itself with these two components and starts sending its camera feed (see Fig: 3). On a side note it's worth noting that the camera feed is first processed through motion detection plug-in to save on bandwidth and time-stamped with data and location. Also the camera feed is stored locally, for redundancy reasons.

And that is the extent of the Ares functionality. It's rather a dull component, more like an eye in a human - the eye by itself cannot do any processing, but the brain does it.

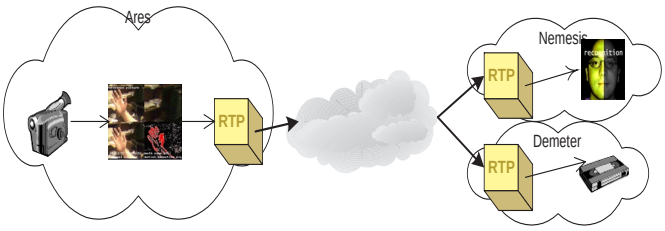

Figure 3: Ares frames passing through the motion detection engine, handled off by media protocol to be distributed to Nemesis (for face recognition) and Demeter (for storage).

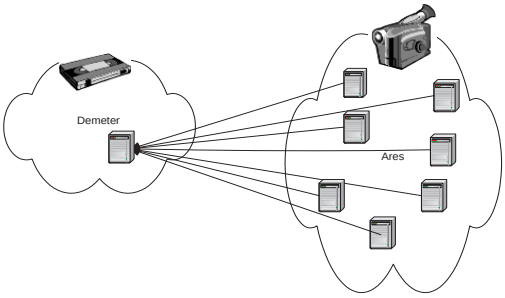

Figure 4: Demeter interaction with multiple systems in Ares

The transportation mechanism implemented in the work to transfer the camera feed was RTP multicast. More information about RTP is found in the "RFC 1889".

It is also modular - if one of the systems from this pool fails, there are many other ones that can take the job over, thought they might not be in the same physical place.

2.5 Demeter

Demeter - the goddess of harvest - is responsible for storing the camera feed from Ares (see Fig: 4). It's a repository serving to collect in one pool the camera feed from various Ares components. It's intended to

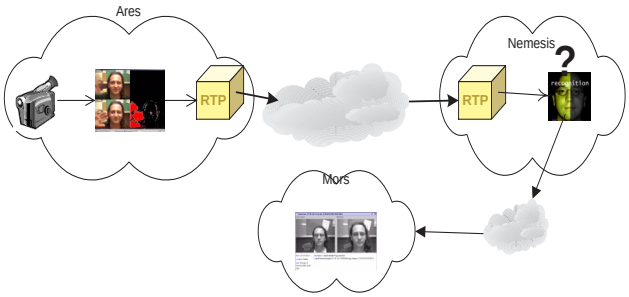

Figure 5: Nemesis interaction with Ares and Mors

Figure 6: Event from Nemesis

allow the generation of time-lapse movies from each individual Ares. This allows for precise replaying the camera feed when a face was recognized from a centralized location.

Since this component requires vast storage capacity, a pool of systems is required. Therefore the Demeter's pool is scalable - as the need increases, more systems are added into the pool. Also machines can be removed and added dynamically (for hardware upgrades for example) from the pool.

In summary, Demeter is a modular piece of the distributed system. This component stores the camera feed (which nota bene is also being done on Ares) - and furthmore provides another functionality: a centralized location to replay the camera feed whenever required. The implementation in the camera-feed is received using RTP.

2.6 Nemesis

Nemesis - the Greek Goddess of vengeance who punished those who had broken the moral code. This component contains the face-recognition engine. It receives the camera feed from Ares and compares the image to the ones in its database (see Fig: 5). Since the face recognition task on a large scale of images is quite computationally intensive, this component requires a hefty pool of high-performance machines (preferable). The images not processed immediately are buffered. This allows for the later matching of faces. When a match occurs, an *event* is sent to Mors along with the image from the camera feed, the matching image, date, and location of match (see Fig: 6). This event is also saved locally for redundancy purpose.

The under-laying face recognition technology is quite computationally intensive . To accommodate high inflow of camera feed among these systems the inflow of data needs to be distributed across the pool. It is also worth noting it is possible to have a number of face recognition engines in the Nemesis component. This would allow for finding a match that intersects the results of the face recognition engines.

The mechanism for finding Mors component is the same as in all other components - by contacting Hermes, the load balancer and finding the least loaded Mors server. The transport mechanism to receive the camera feed is the same as in Demeter - RTP.

In summary, Nemesis is the most second important component of this framework - it does the face recognition and notifies other components about possibly matches.

2.7 Mors

Mors - (aka Thanatos) - personification of death - is the component that receives face recognition events. This component is responsible for showing the operator (the user) that a face-recognition match between an image in the Nemesis face database and the camera feed from Ares occurred. The location of the camera, along with date, the camera image and the matched image is submitted to Mors.

In summary, Mors provides a centralized pool

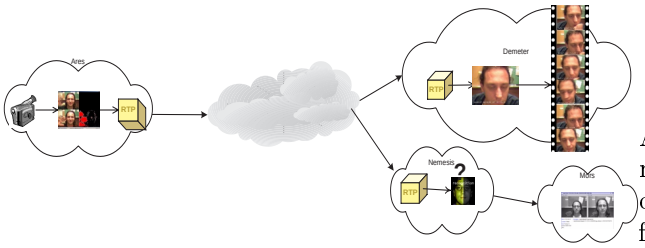

Figure 7: Primary operation of Apollo

where events are recorded. Having a single point (or many single points) where events are stored facilities the instant comparison of the matches with the camera's feed. This allows for humans to verify the result and act accordingly if there is a need.

2.8 How they work together

Each system is autonomous - in a component pool each machine is completely independent of each other. However, not each pool is independent of each other. Nemesis communicates with Hermes and Mors. Ares communicates with Hermes, Nemesis and Demeter.

Hermes, being the "police officer" is the most critical component. Without Hermes presence, Ares would be unable to find its required components and newly started services would be unable to find their required services. Thought components that already have found their services and are communicating are not affected (unless one of the components fails and its necessary to find an replacement).

2.9 Primary operation of Apollo

There are many operations in this system. Each component by itself performs internally many functions. This section focuses on the primary operations of a face recognition system (see Fig: 7, which are:

- How a frame captured by the camera traverses through the system.
- What happens to it when it matches to the internal database of images?
- How is the camera feed saved on an off-line storage?

The camera feed is handled by one component - Ares, Ares receives the camera feed and based on motion detection engine either passes the frames on or drops them. If the frame passed the check, the frame is sent to the next two components - Demeter and Nemesis. When Demeter and Nemesis received it, they saved it locally. It's assumed that Ares has already obtained the address of Nemesis and Demeter from Hermes or individually.

Demeter stores the frames until a predefined amount of time has elapsed (usually twenty four hours) and at which point it generates a movie from the stored frames and deletes the frames.

Nemesis buffers the received frames and at some predefined period of time tries to match the frames to its internal database of pictures. If there is no match the frame is expunged. Otherwise an event (see Fig: 6) is generated which is saved locally and propagated to Mors. Mors upon receiving the message saves it locally and displays the event to the operator.

3 Dynamic aspects

The agility and adaptability is a requirement for a scalar distributed system. The system must be flexible and capable of addressing various problems: network loss connectivity, power outage, demand for more components, notification of new services and switching new load onto them, and many more. All of this must be handled in a reliable distributed system. These are a must for a true distributed system.

3.1 Interactions of a distributed system

The most essential interactions in this distributed system are:

- Notification and registration of new components.
- Querying the components for its load and availability.
- Finding components with the least load.

- Discontinuing the use of deceased components and using new ones.
- Work dynamically.

3.2 Notification and registration of new components

Each component, except Aries, whenever they are started notifies the other components about its presence. The only component that takes notice of it this is Hermes. Hermes, being the load balancer needs to know the whole set of network components.

The method by which the components find out about each other presence is by using Jini technology - mainly leveraging the multicast request protocol as described in Jini(TM) Architecture Specifications [?]. The method by which a component announces its presence is by locating the lookup services (which are native to Jini), download code to control the lookup service, use that code to register itself (and also upload its own code) and then periodically renew the registration. The code that is uploaded includes simple information that can be modified and queried - mainly the count of users, the maximum amount of users that can be handled, and the address of the system. This information is used in finding the load and availability of the system, which is explained in the next section.

All of these components: Hermes, Mors, Nemesis and Demeter are services (in Jini terminology), while Ares is the client. All the services are using Jini to announce their presence and find, if needed, the other components (Nemesis looks for Mors using Hermes' knowledge). Ares on the other hand, being a client doesn't announce its presence - it searches for the services it requires.

3.3 Querying the components for its load and availability

After the service components have been registered with the lookup server, Hermes queries each new found components for its information (see Fig 8). It does that every predefined amount of time. This allows for retrieval of near-real time statistical information on the load of each service. It also allows for discovering if the service has been disconnected or is no longer operational and accordingly purge information about the service.

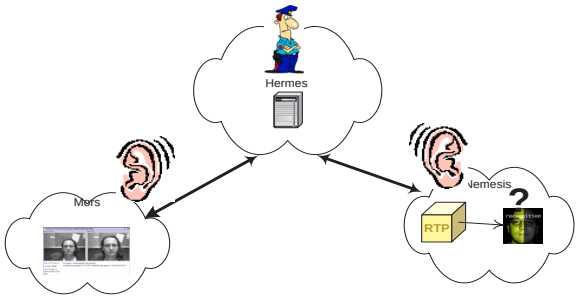

Figure 8: Information querying by Hermes

Only Hermes queries for these information. All other components just provide the pertinent information and change their information accordingly to their status.

3.4 Finding components with the least load

Ares and Nemesis are two of the components that require access to the other components. Ares requires Demeter and Nemesis, while Nemesis requires Mors. Each of these clients needs to find the appropriate service.

The requesting client queries Hermes, which knows the least populated service in the desired pool. Hermes provides the address to the least loaded service and the requesting client uses that address to talk to the service directly. If the requested component is not available, no address is returned.

It is assumed to that the components can and will stop working at some point. Therefore the connection between the components can break at any time and should be re-established. If there are no desired components at the current time then the service should continue asking Hermes for that component repeatedly until its found.

When the required component is found it's address is cached and periodically checked. This makes it

possible to discover dead services and request new ones from Hermes. Vice-versa - if the connection is ok, there is no need query Hermes for a least loaded service in the pool

3.5 Work dynamically

With the idea of notification, registration, checking the components its feasible to adjust to changing network conditions. New services can be taken advantage of and other nodes in a pool can be shutdown for maintenance. All these features allows for flexible rollover off services. In turn making the whole system capable of working truly dynamic scalable distributed fashion.

4 Future work

Face recognition has many advantages in today's life. Understanding the under laying mechanism of human's face recognition could be used to build computer systems capable of recognizing human faces. The application of such system spawns many topics: surveillance systems, biomedical field, military, system security, and many more. With so many applications a methodology to match faces in near real-time is an urgent need. This work presents a distributed scalable face recognition framework that can be used to solve computational intense problems.

This work has presented a novel idea of matching faces in face recognition technology. The technology of face-recognition fused with distributed computing presents a clear working example of implementing on a large scale a face-recognition suite.

The work has been implemented and shows to operate in near-real time environment. However, there are improvements to be made.

Using more than one face recognition technology to recognize a face, and using the intersecting results to derive the matching face is the first improvement. Complementing one technique with another technique to improve the face recognition matching is surely a priority number one.

The next improvement can be made in the motion detection engine. Optimization to it, such as scaling the frame to a lower size, only analyzing every fifth frame, and getting rid of masking the bits.

Another important improvement is to provide support for other thin-client recognition modules. It can be such technologies as iris-recognition, fingerprint-recognition, or even speech recognition. The combination of many recognition technologies can serve to identify a person more easily.

Future work also requires more research to be done in face operations. Automatically removing background information from a frame to not introduce any unnecessary artifacts. Generating from one face a multitude of others with different alternations - beard (or the lack of it); sunglasses; long hair (or short hair). All of these would serve to identify a possible match with a person who might have change his or her face appearance.

References

[1] N.M. Allinson, A.W. Ellis, B.M. Flude, and A.J Luckman. A connectionist model of familiar face recognition. In *IEE Colloquium on Machine Storage and Recognition of Faces*, pages 1–10, 1992. Digest No: 1992/017.

[2] R. Brunelli and T. Poggio. Caricatural effects in automated face perception.

[3] R. Brunelli and T. Poggio. Face recognition through geometrical features.

[4] R. Brunelli and T. Poggio. Hyberbf networks for gender classification.

[5] R. Brunelli and T. Poggio. Hyperbf networks for real object recognition. In *Proc. of the 12th IJCAI*, pages 1278–1284, Sidney, Australia, 1991.

[6] Kenneth R. Castleman. *Digital Image Processing*. Prentice-Hall, Inc., 1996.

[7] Ross Cutler. Face recognition using infrared images and eigenfaces. April 1996.

[8] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics*

Principle and Practice. Addison-Wesley, Inc., 1997.

[9] Francis Galton. *Personal Identification And Description*. June 1888.

[10] Gaston Gonnet. Singular value decomposition and eigenvalue decomposition. November 2001.

[11] Audio-Video Transport Working Group, H. Schulzrinne, GMD Fokus, S. Casner, Precept Software Inc., R. Frederick, Xerox Palo Alto Research Center, V. Jacobson, and Lawrence Berkeley National Laboratory. Rtp: A transport protocol for real-time applications. January 1996.

[12] William H., Teukolsky Saul A., Vetterling William T., and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992.

[13] L.D. Harmon, M.K. Khan, R. Lasch, and P.F. Ramig. Machine identification of human faces. *Pattern Recognition*, 1981.

[14] Jim Hefferon. *Linear Algebra*.

[15] T. Kanade. Computer recognition of human faces. *Interdisciplinary Systems Research*, 1977. Birkhauser Verlag.

[16] Mohamed Amine Khamsi. Eigenvalues and eigenvectors technique.

[17] T. Kohonen. *Self-organization and associative memory*. Springer-Verlag, 1988. 2nd Edition.

[18] Sun Microsystem. *Java(TM) Remote Method Invocation Specification*. 1998.

[19] Sun Microsystems. *JMF Frequently Asked Questions*.

[20] Sun Microsystems. *Java(TM) Media Framework API Guide*. November 1999.

[21] Sun Microsystems. *Jini Network Technology Datasheet*. May 2001.

[22] Sun microsystems. *Jini(TM) Architecture Specification*. Sun Microsystems, 2001.

[23] L. Najman, R. Vaillan, and E. Pernot. Face from sideview to identification. In G. Vernazza, A.N. Venetsanopouls, and C. Braccini, editors, *Image Processing: Theory and Applications*. Elsevier Science Publishers, 1993.

[24] O. Nakamura, S. Mathur, and T. Minami. Identification of human faces based on isodensity maps. *Pattern Recognition*, pages 263–272, 1991.

[25] Alexander Pentland and Terrence Sejnowski. Neural networks and eigenfaces for finding an analyzing faces.

[26] A. Samal and P.A. Lyengar. Automatic recognition and analysis of human faces and facial expressions: A survey. *Pattern Recognition*, 1992.

[27] M. Turk and A. Pentland. Eigenfaces for recognition. In *Journal of Cognitive Neuroscience*, March 1991.

[28] Matthew A. Turk and Alex P. Pentland. Face recognition using eigenfaces. May 1991.

[29] K.H. Wong, H.H.M. Law, and P.W.M. Tsang. A system for recognising human faces. In *Proceedings of the Internation Conference on Acoustics, Speech and Signal Processing*, pages 1638–1642, 1989.

[30] C.J. Wu and J.S. Huang. Human face profile recognition by computer. *Pattern Recognition*, pages 255–259, 1990.