**UNIVERSIDAD DON BOSCO**

**SOYAPANGO**


TRABAJO DE GRADUACIÓN PREPARADO PARA LA

**FACULTAD DE INGENIERIA**

# Investigación y desarrollo dè un Prototipo de Sistema Biométrico para la identificación de Rostros Humanos utilizando la Teoría de los Eigenvectores y Eigenvalores

PARA OPTAR AL GRADO DE

**INGENIERO EN CIENCIAS DE LA COMPUTACIÓN**

**PRESENTADO POR**

Rafael Ernesto Morán Bautista

Remberto Napoleón Díaz Jovel


**ASESOR**

Ing. Jaime Antonio Anaya

FECHA: Marzo 15, 2007

SAN SALVADOR, EL SALVADOR, C. A

UNIVERSIDAD DON BOSCO

FACULTAD DE INGENIERÍA

ESCUELA DE COMPUTACIÓN

AUTORIDADES:

RECTOR:

ING. FEDERICO MIGUEL HUGUET RIVERA

VICERECTOR:

PBRO. VÍCTOR BERMÚDEZ YANEZ, SDB

SECRETARIO GENERAL:

LIC. MARIO RAFAEL OLMOS ARGUETA

DECANO DE LA FACULTAD DE INGENIERÍA:

ING. ERNESTO GODOFREDO GIRÓN

DIRECTOR DE ESCUELA DE COMPUTACIÓN:

LIC. JORGE MAURICIO COTO

ASESOR DEL TRABAJO DE GRADUACIÓN:

ING. JAIME ANTONIO ANAYA

JURADO EVALUADOR:

ING. HERNÁN AREVALO

ING. ALBERTO DÁVILA

ING. CARLOS HÉRCULES

UNIVERSIDAD DON BOSCO

FACULTAD DE INGENIERÍA

ESCUELA DE COMPUTACIÓN

SUBCOMITE EVALUADOR ASESOR TUTOR

JURADO EVALUADOR DEL TRABAJO DE GRADUACIÓN

Ing. Jaime Antonio Anaya

**ASESOR**

Ing. Carlos Tejada

**TUTOR**

Ing. Hernán Arevalo

**JURADO**

Ing. Alberto Dávila

**JURADO**

Ing. Carlos Hércules

**JURADO**

# AGRADECIMIENTOS

A Dios, dador de vida de mis seres amados.
Mi madre, Marina; fortaleza, amor y templanza
Mi esposa, Nancy; humildad, sencillez y pureza
Mi padre, Napoleón; inspiración, ideal y carácter
Mis amigos, hermanos y compañeros; amistad, alegría y ánimo.
Mi gran amigo Rafael Morán; carácter, sencillez y fortaleza.

Napoleón Díaz Jovel

Gracias a Dios y a María; quienes me han permitido llegar a esta etapa de la vida, y me enseñaron que las pruebas y dificultades sirven para forjar el carácter, despreciar el orgullo y ser agradecido por las oportunidades exitosas como por las situaciones frustrantes.

A mis seres queridos y cercanos quienes siempre me rodearon de palabras de ánimo, por su apoyo y su tolerancia; sobre todo en mis momentos de desdén y orgullo.

Rafael Ernesto Morán Bautista

# INDICE

# INDICE DE TABLAS Y FIGURAS

## Figuras

## Tablas

## SUMARIO

La identificación biométrica es la verificación de la identidad de una persona basado en características de su cuerpo o su comportamiento; por ejemplo su mano, el iris de su ojo, su voz, su firma, sus huellas dactilares o su rostro.

De los sistemas biométricos en general; quizás el más ampliamente implementado, debido a la relación exactitud-disponibilidad-costo ha sido el de análisis de huellas dactilares; sin embargo existen otras alternativas, como por ejemplo, el reconocimiento del iris y el reconocimiento facial los cuales ofrecen similar exactitud y precisión. Por otro lado el factor costo/beneficio va mejorando proporcionalmente al desarrollo de la tecnología.

Para los seres humanos, el sistema de identificación desarrollado y utilizado por excelencia es el reconocimiento facial; el cerebro humano es capaz de identificar miles de rostros los cuales ha observado a través de su vida; esta familiaridad en los rostros persiste aun después de varios años de separación. El sistema humano de reconocimiento de rostros es tan avanzado que permite la identificación de personas a pesar de existir otros estímulos visuales y la variación de los rostros debido al uso de otros accesorios (por ejemplo lentes, gorros, maquillaje, corte de pelo y barba) o de la edad de las personas.

Los modelos computacionales dedicados al reconocimiento facial aun están en desarrollo, sin embargo el perfeccionamiento de los mismos puede hacer muchas contribuciones en el ámbito teórico, psicológico y práctico a través del desarrollo de sistemas de seguridad e identificación.

# TERMINOS Y LICENCIAS

El presente trabajo de investigación consta de 2 partes importantes y relacionadas; la documentación y la herramienta construída en lenguaje Python; con el objetivo de garantizar el acceso a estos dos recursos para investigaciones posteriores se incluye el siguiente esquema de licenciamiento:

- La documentación se encuentra protegida bajo la licencia de documentación Libre de GNU, lo cual garantiza el uso de esta documentación con copyleft[1].
- El software desarrollado en Python se encuentra protegido bajo la licencia Artística 2.0, esta licencia es para software libre compatible con la GNU GPL[2].

A continuación se transcribe el detalle de cada una de las licencias:

Licencia de la documentación:

```
GNU Free Documentation License
                Version 1.2, November 2002


 Copyright (C) 2000,2001,2002  Universidad Don Bosco, Napoleón Díaz Jovel,
 Rafael Morán. Canton Venecia, Soyapango El Salvador Centroamérica.
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.


0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other
functional and useful document "free" in the sense of freedom: to
assure everyone the effective freedom to copy and redistribute it,
with or without modifying it, either commercially or noncommercially.
```

[1] http://www.gnu.org/copyleft/fdl.es.html

[2] http://dev.perl.org/perl6/rfc/346.html

Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept

compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section

of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

## 8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

## 9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

## 10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See

http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

## Licencia de Software:

The Artistic License
Version 2.0beta4, October 2000



### Preamble

This copyright license states the terms under which a given free software Package may be copied, modified and/or redistributed, while the Originator(s) maintain some artistic control over the future development of that Package (at least as much artistic control as can be given under copyright law while still making the Package open source and free software).

This license is bound by copyright law, and thus it legally applies only to works which the copyright holder has permitted copying, distribution or modification under the terms of the Artistic License, Version 2.0.

You are reminded that You are always permitted to make arrangements wholly outside of a given copyright license directly with the copyright holder(s) of a given Package. If the terms of this license impede your ability to make full use of the Package, You are encouraged to contact the copyright holder(s) and seek a different licensing arrangement.

### Definitions

"Package" refers to the collection of files distributed by the Originator(s), and derivatives of that collection of files created through textual modification.

"Standard Version" refers to the Package if it has not been modified, or has been modified only in ways suggested by the Originator(s).

"Modified Version" refers to the Package, if it has been changed by You via textual modification of the source code, and such changes were not suggested by the Originator(s).

"Originator" refers to the author(s) and/or copyright holder(s) of the

Standard Version of the Package.

"You" and "Your" refers to any person who would like to copy, distribute, or modify the Package.

"Distribution Fee" is any fee that You charge for providing a copy of this Package to another party. It does not refer to licensing fees.

"Freely Available" means that:

(a) no fee is charged for the right to use the item (though a Distribution Fee may be charged).

(b) recipients of the item may redistribute it under the same conditions they received it.

(c) If the item is a binary, object code, bytecode, the complete corresponding machine-readable source code is included with the item.

Permission for Use and Modification Without Redistribution

(1) You are permitted to use the Standard Version and create and use Modified Versions for any purpose without restriction, provided that you do not redistribute the Modified Version to others outside of your company or organization.

Permissions for Redistribution of the Standard Version

(2) You may make available verbatim copies of the source code of the Standard Version of this Package in any medium without restriction, either gratis or for a Distribution Fee, provided that you duplicate all of the original copyright notices and associated disclaimers. At Your discretion, such verbatim copies may or may not include compiled bytecode, object code or binary versions of the corresponding source code in the same medium.

(3) You may apply any bug fixes, portability changes, and other modifications made available from any of the Originator(s). The resulting modified Package will still be considered the Standard Version, and may be copied, modified and redistributed under the terms
of the original license of the Standard Version as if it were the Standard Version.

Permissions for Redistribution of Modified Versions of the Package as Source

(4) You may modify your copy of the source code of this Package in any way and distribute that Modified Version (either gratis or for a Distribution Fee, and with or without a corresponding binary, bytecode
or object code version of the Modified Version) provided that You

clearly indicate what changes You made to the Package, and provided that You do at least ONE of the following:

(a) make the Modified Version available to the Originator(s) of the Standard Version, under the exact license of the Standard Version, so that the Originator(s) may include your Modifications into the Standard Version (at their discretion).

(b) modify any installation scripts and procedures so that installation of the Modified Version will never conflict with an installation of the Standard Version, include for each program installed by the Modified Version clear documentation describing how it differs from the Standard Version, and rename your Modified Version so that the name is substantially different from the Standard Version.

(c) permit and encourage anyone who receives a copy of the Modified Version permission to make your modifications Freely Available in some specific way.

If Your Modified Version is in turn derived from a Modified Version made by a third party, then You are still required to ensure that Your modified Version complies with the requirements of this license.

Permissions for Redistribution of Non-Source Versions of Package

(5) You may distribute binary, object code, bytecode or other non-source versions of the Standard Version of the Package, provided that you include complete instructions on where to get the source code of the Standard Version. Such instructions must be valid at the time of your distribution. If these instructions, at any time while You are carrying our such distribution, become invalid, you must provide new instructions on demand or cease further distribution. If You cease distribution within thirty days after You become aware that the instructions are invalid, then You do not forfeit any of Your rights under this license.

(6) You may distribute binary, object code, bytecode or other non-source versions of a Modified Version provided that You do at least ONE of the following:

(a) include a copy of the corresponding source code for the Modified Version under the terms indicated in (4).

(b) ensure that the installation of Your non-source Modified Version does not conflict in any way with an installation of the Standard Version, include for each program installed by the Modified Version clear documentation describing how it differs from the Standard Version, and rename your Modified Version so that the name is substantially different from the Standard Version.

(c) ensure that the Modified Version includes notification of the changes made from the Standard Version, and offer to provide machine-readable source code (under a license that permits making that source code Freely Available) of the Modified

Version via mail order.

Permissions for Inclusion of the Package in Aggregate Works

(7) You may aggregate this Package (either the Standard Version or Modified Version) with other packages and distribute the resulting aggregation provided that You do not charge a licensing fee for the Package. Distribution Fees are permitted, and licensing fees for other packages in the aggregation are permitted. Your permission to distribute Standard or Modified Versions of the Package is still subject to the other terms set forth in other sections of this license.

(8) In addition to the permissions given elsewhere by this license, You are also permitted to link Modified and Standard Versions of this Package with other works and distribute the result without restriction, provided You have produced binary program(s) that do not overtly expose the interfaces of the Package. This includes permission to embed the Package in a larger work of your own without exposing a direct interface to the Package. This also includes permission to build stand-alone binary or bytecode versions of your scripts that require the Package, but do not otherwise give the casual user direct access to the Package itself.

Items That are Never Considered Part of a Modified Version Package

(9) Works (including, but not limited to, subroutines and scripts) that you have linked or aggregated with the Package that merely extend or make use of the Package, but are not intended to cause the Package to operate differently from the Standard Version, do not, by themselves, cause the Package to be a Modified Version. In addition, such works are not considered parts of the Package itself, and are not bound by the terms of the Package's license.

Acceptance of License and Disclaimer of Warranty

(10) You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to copy, modify or distribute the Standard or Modified Versions of the package. These actions are prohibited by copyright law if you do not accept this License. Therefore, by copying, modifying or distributing Standard and Modified Versions of the Package, you indicate your acceptance of the license of the Package.

(11) Disclaimer of Warranty:

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT UNLESS REQUIRED BY LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER OR CONTRIBUTOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,

PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

# 1. INTRODUCCIÓN

Los sistemas biométricos se basan en características o rasgos físicos cuantificables ó personales de comportamiento, los cuáles son usados para reconocer o verificar la identidad de una persona a través de medios automáticos.

El reconocimiento facial es un área de estudio de la biométrica el cual pretende el desarrollo de modelos computacionales que efectúen la identificación de rostros humanos de una forma similar y efectiva a como lo realizan los seres humanos entre sí.

Los sistemas biométricos han sido un área importante de investigación en los años recientes debido a las siguientes características:

- Universalidad, lo cual significa que cada persona debe de tener esas características[3].
- Unicidad, lo cual significa que dos personas no deben de ser la misma en términos de las características.
- Permanencia, lo cual indica que las características deben ser invariantes con el tiempo.
- Colectibilidad, lo cuál indica que las características pueden ser medibles cuantitativamente.

[3] No aplica en el caso del rostro facial debido a la notable similitud de rostros entre gemelos univitelinos o monocigóticos que provienen de la fertilización de un solo óvulo; sin embargo estos casos representan menos del 1/50 de probabilidad de que ocurran; sin embargo existen técnicas de análisis facial en 3 dimensiones las cuales pueden diferenciar inclusive entre gemelos idénticos.

## *1.1 Introducción a Sistemas biométricos como mecanismos de control de Acceso.*

Los sistemas tradicionales utilizados en el control de acceso a recintos se basan en los sistemas de tarjetas magnéticas, sistemas de tarjetas con código de barras, sistemas de captura de clave o combinación de ellos. Estos sistemas involucran el uso de una tarjeta que hay que llevar siempre consigo y la cual no está exenta de perderse, dañarse, ser robada o falsificada, con lo cual la seguridad del recinto se hace más vulnerable a fallas. Por esta razón, si se cuenta con un sistema más robusto y de mayor confiabilidad se pueden evitar los problemas antes mencionados.

Cuando la seguridad de acceso es muy importante; es común encontrar algún tipo de sistemas biométrico que determinen la identidad del visitante; debido a la facilidad de implementación y el costo relativamente bajo; es muy común la existencia de sistemas biométricos basados en el tamaño de las manos o en las huellas dactilares; sin embargo existen implementaciones más costosas pero más fidedignas como lo es el escaneo de retina o del iris; y probablemente existan mecanismos de reconocimiento facial sin embargo aun existe mucha investigación sobre las mejores técnicas para implementarlo.

La exactitud del reconocimiento facial depende de la implementación que se utilice y es directamente proporcional al costo; los modelos más exactos generalmente involucran un reconocimiento en 3 dimensiones del rostro humano los cuales son valorados junto con el reconocimiento del iris como mecanismos de identificación con índices de error muy bajos (algunos

vendedores estiman un FAR[4] del 0% y FRR[5] del 3.1%) por lo tanto los más fidedignos.

### *1.2 Funcionamiento de un sistema biométrico*

El funcionamiento de un sistema biométrico consiste en el registro de una o más características de una persona en el sistema; dichas características son cuantificadas y almacenadas en una base de datos.

Cuando se desea verificar la identidad de una persona, esta proporciona nuevamente las características físicas al sistema las cuales deben cuantificarse nuevamente y compararse contra el valor almacenado originalmente en la base de datos; la comparación no es estrictamente igual, incluye un factor adecuado de tolerancia de error entre las características almacenadas; este factor de tolerancia se debe principalmente al ruido que siempre está implícito en las señales digitales y otras variables circunstanciales (medio ambiente, posición o estado de las características físicas medibles por el sistema biométrico, por mencionar algunas) que determinan diferencias entre la nueva medida biométrica y la medida inicial almacenada en la base de datos.

Los sistemas biométricos siempre trabajan en base a 3 términos definidos como tasa de falso positivo (False Aceptante Rate o FAR), la tasa de falso negativo (False Non match Rate o FNMR y el fallo de tasa de alistamiento (Failure to enroll Rate, FTR o FER).

---

[4] False Acceptance Rate: Porcentaje de incidentes falsos que fueron aceptados como válidos.

[5] False Rejection Rate: Porcentaje de incidentes verdaderos que fueron rechazados como inválidos.

Los factores FAR y FRR varían según la tolerancia de error admitida por el sistema; la convergencia entre estos dos factores constituye la tasa de error igual (Equal Error Rate o EER) conocida también como la tarifa de error de cruce (Crossover Error Rate o CER). Lo deseable en lo que se refiere a rendimiento de un sistema biométrico es que la EER o CER se aproxime a 0; visualmente la CER se puede graficar como el punto de convergencia entre FAR y FRR tal como lo ilustra la Figura 1.

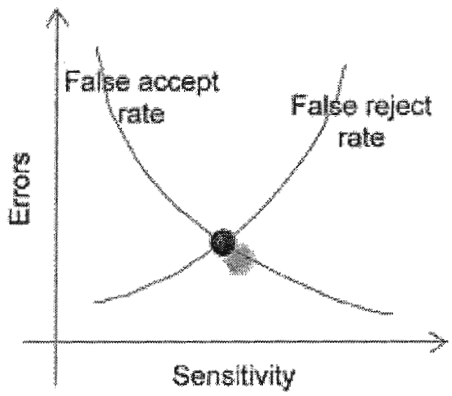



*Figura 1: FAR, FRR, CER*

### *1.3 Ventajas del reconocimiento facial sobre otras técnicas biométricas.*

- Es una técnica no intrusiva, es decir, no requiere una interacción física con el usuario; esto crea una disponibilidad del usuario hacia el uso de la tecnología.
- Es muy precisa y presenta un elevado rendimiento en el proceso de enrolamiento y verificación.
- No requiere la presencia de un experto para interpretar los resultados de las comparaciones.

- Puede utilizarse en el hardware actual de cámaras de vigilancias sin incurrir en costo de equipos avanzados y/o especializados.[6]
- Se pueden utilizar imágenes existentes para efectuar el enrolamiento; no es necesario en algunos casos la captura inicial de imágenes.
- Es el único sistema de biometría que permite la identificación pasiva en muchos entornos; por ejemplo la identificación de terroristas o personas buscadas por la ley en un aeropuerto, centros comerciales, lugares turísticos por mencionar algunos ejemplos.
- No existen problemas de daño en la epidermis del rostro facial, tal como ocurre en algunos casos de huellas dactilares, donde por efectos de ácidos o trabajo manual las huellas dactilares son ilegibles o no válidas para un enrolamiento o identificación.
- Como un sistema biométrico no intrusivo, es higiénico y no requiere mayor mantenimiento.
- Los costos son menores, si se implementa el modelo en 2 dimensiones ya que se reutiliza hardware existente o se compran equipos de vigilancia normal.

## *1.4 Aplicaciones de la biometría facial.*

La técnica de la biometría facial esta siendo evaluada y adoptada como un mecanismo válido y preciso para la identificación de personas; para mencionar ejemplos reales sobre la adopción de esta tecnología se muestran los siguientes:

---

[6] Se requiere hardware especializado cuando el reconocimiento facial es realizado utilizando modelado en tres dimensiones.

- Los Estados miembros de la Unión Europea decidieron desde diciembre 2004 incluir datos biométricos (imágenes faciales y huellas dactilares) en los pasaportes a través de chips internos.

- La Ley Patriota de los EEUU y la Ley Acta de Seguridad Incrementada de Fronteras y Reforma de Visas de Ingreso, solicitaron al Instituto Nacional de Normas y Tecnología (NIST por sus iniciales en inglés) y otros organismos que desarrollen y certifiquen estándares de tecnología para los sistemas de control de visas. NIST consideró la tecnología de reconocimiento de iris como una candidata prometedora pero optó por un enfoque de combinación de huellas digitales e imágenes faciales, concluyendo que las imágenes faciales proporcionan la misma precisión en la verificación que las huellas digitales.

- Reconocimiento de rostros en terminales móviles. Los terminales móviles de tercera generación llevan integrados dispositivos de captación de imagen. Esta característica permite el uso del reconocimiento de rostro para autentificar a los usuarios.

# 2. ANTECEDENTES

La biometría no se puso en práctica en las culturas occidentales hasta finales del siglo XIX, pero era utilizada en China desde al menos el siglo XIV. Un explorador y escritor que respondía al nombre de Joao de Barros, escribió que los comerciantes chinos estampaban las impresiones y las huellas de la palma de las manos de los niños en papel con tinta. Los comerciantes hacían esto como método para distinguir entre los niños jóvenes.

En Occidente, la identificación confiaba simplemente en la "memoria fotográfica", hasta que Alphonse Bertillon, jefe del departamento fotográfico de la Policía de París, desarrolló el sistema antropométrico (también conocido más tarde como Bertillonage) en 1883. Éste era el primer sistema preciso, ampliamente utilizado científicamente para identificar a criminales y convirtió a la biométrica en un campo de estudio. Funcionaba midiendo de forma precisa ciertas longitudes y anchuras de la cabeza y del cuerpo, así como registrando marcas individuales como tatuajes y cicatrices.

El sistema de Bertillon fue adoptado extensamente en occidente hasta que aparecieron defectos en el sistema principalmente problemas con métodos distintos de medidas y cambios de medida. Después de esto, las fuerzas policiales occidentales comenzaron a usar la huella dactilar esencialmente el mismo sistema visto en China cientos de años antes.

En estos últimos años la biometría ha crecido desde usar simplemente la huella dactilar, a emplear muchos métodos distintos teniendo en cuenta varias medidas físicas y de comportamiento. Las aplicaciones de la biometría también han aumentado desde sólo identificación hasta sistemas de seguridad y más.

Debido a la importancia y atracción de temas relacionados con la visión artificial y reconocimiento de imágenes, las universidades en El Salvador, específicamente la Universidad Don Bosco, está impulsando la investigación de este tipo de proyectos; para mencionar unos ejemplos, ya existen trabajos de investigación y prototipos que identifican señas del lenguaje sordomudo, prototipos que estudian el reconocimiento de huellas dactilares y en este caso particular el reconocimiento facial aplicando conceptos de algebra lineal.

## *2.1 Visión artificial*

La Visión artificial, también conocida como Visión por Computador (del inglés Computer Vision) o Visión técnica, es un subcampo de la inteligencia artificial. El propósito de la visión artificial es programar un computador para que "entienda" una escena o las características de una imagen.

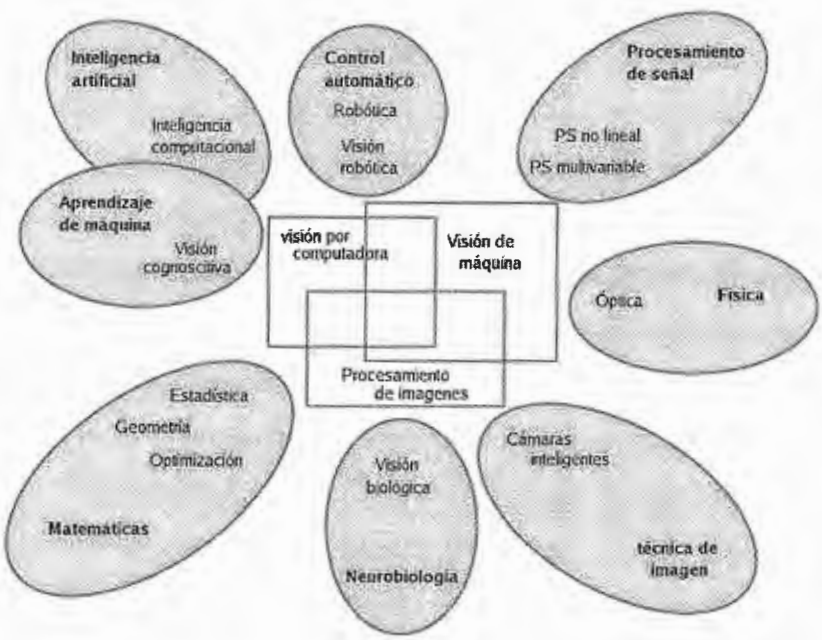



*Figura 2: Esquema de relaciones entre visión por computadora y otras áreas afines*

El campo de la visión artificial es muy amplio y comúnmente entrelazado con otras áreas de investigación de la informática, matemática, biología, robótica, por mencionar algunos ejemplos; la Figura 2 esquematiza dichas relaciones.

Los objetivos típicos de la visión artificial incluyen:

- La detección, segmentación, localización y reconocimiento de ciertos objetos en imágenes (por ejemplo, caras humanas).
- La evaluación de los resultados (ej.: segmentación, registro).
- Registro de diferentes imágenes de una misma escena u objeto, i.e., hacer concordar un mismo objeto en diversas imágenes.
- Seguimiento de un objeto en una secuencia de imágenes.
- Mapeo de una escena para generar un modelo tridimensional de la escena; tal modelo podría ser usado por un robot para navegar por la escena.
- Estimación de las posturas tridimensionales de humanos.
- Búsqueda de imágenes digitales por su contenido.

Estos objetivos se consiguen por medio de reconocimiento de patrones, aprendizaje estadístico, geometría de proyección, procesado de imágenes, teoría de grafos y otros campos. La visión artificial cognitiva está muy relacionada con la psicología cognitiva y la computación biológica. El prototipo a desarrollar basa sus cimientos de análisis y diseño, en la Visión Artificial, sus conceptos, algoritmos matemáticos, algoritmos de algebra lineal, técnicas de procesamiento y tratamiento funcional de señales ruido e imágenes.

## *2.2 Definición de Sistemas Biométricos*

Los Sistemas biométricos de identificación son aquellos que permiten reconocer una persona basándose en características fisiológicas o de comportamiento. Se caracterizan por la necesidad de que la persona este físicamente presente en el lugar de la identificación, pueden o no requerir la colaboración del usuario e incluso pueden obviar la necesidad de que este conozca la existencia del sistema que lo esta identificando.

### 2.2.1 Características biométricas cuantificables

Los sistemas biométricos se pueden dividir en 2 ramas, dependiendo del tipo de característica biométrica que utilizan para la identificación y/o verificación de las personas:

- Sistemas que cuantifican una característica biométrica de tipo fisiológica
- Sistemas que cuantifican una característica biométrica de comportamiento.

Las características biométricas fisiológicas están relacionadas con rasgos físicos universales medibles en los seres humanos; los métodos de tipo fisiológico incluyen los siguientes: reconocimiento de huellas dactilares, exploración del iris, exploración de la retina, geometría de la mano, reconocimiento facial en luz visible (2D ó 3D), reconocimiento de la imagen tomográfica facial, análisis de ADN, reconocimiento auricular, exploración del patrón venoso en la muñeca, etc.

Las características biométricas de comportamiento no están relacionadas con rasgos físicos sino con la forma diversa aprendizaje y ejecución de ciertas actividades humanas; entre los métodos basados en

comportamientos tenemos: identificación por la voz, identificación por la escritura, dinámica de pulsación en teclado, análisis del patrón de marcha, etc.

### 2.2.2 Modos de Funcionamiento

Los sistemas biométricos son utilizados (según la característica biométrica de estudio y la madurez de la tecnología) para efectuar 2 tipos de tareas las cuales son las siguientes:

- Verificación
- Identificación

#### *2.2.2.1 Verificación*

En el modo de Verificación, el usuario se identifica mediante un método típicamente no biométrico, como un código (PIN) ó una tarjeta, y el sistema ha de comprobar (verificar) que la identidad proporcionada se corresponde con la realidad.

Son aplicaciones típicas de verificación las siguientes: control de acceso a un recinto, control de acceso a un sistema informático, control de identidad por las autoridades, identificación en votaciones, utilización de servicios (cajeros automáticos, transporte publico, etc.), cobro de servicios (comercio electrónico, pago a distancia, etc.), utilización de dispositivos (teléfonos móviles, automóviles, etc.), confirmación forense de la identidad (identificación de cadáveres, paternidad, etc.).

*2.2.2.2 Identificación*

En el modo de Identificación, se trata de averiguar la identidad del sujeto buscando en una base de datos una representación de parámetros biométricos suficientemente aproximada.

Entre las aplicaciones de identificación se incluyen: identificación forense de huellas dactilares latentes, detección de sujetos en "listas negras" (terrorismo, delincuencia, etc.) en espacios públicos, control de fronteras, cobro automático sin interacción del usuario (montos pequeños).

## 2.2.3 Seguridad

La seguridad de un sistema de acceso basado en palabra de paso o número de identificación personal se basa en la confidencialidad de esa palabra o número; en el caso de una llave o tarjeta de identificación, en evitar su perdida o su duplicación clandestina. Pero en todos estos casos, la introducción del código o el dispositivo físico siempre sin excepción (salvo fallo del sistema) resulta en un acceso garantizado al servicio requerido.

Sin embargo, en los sistemas biométricos, debido a la variabilidad de la información procesada (imagen de una huella, de una cara, medidas de longitud de los dedos, etc.) se pueden dar casos de falso rechazo del usuario legitimo o, lo que es peor, falsa aceptación de un sujeto no autorizado.

En la practica, se plantea un compromiso entre la comodidad del usuario (cada falso rechazo implica un nuevo intento por parte del sujeto que intenta acceder o una alarma innecesaria) y la seguridad del sistema. Cuanta mas similitud se exige entre los parámetros leídos y los almacenados, mas seguridad se obtendrá (menos falsas aceptaciones), pero también mas frecuentes serán los falsos rechazos. Así pues, siempre dispondremos de un

umbral, normalmente ajustable, que nos permita aumentar la seguridad a costa de disminuir la comodidad del usuario.

La probabilidad de falsa aceptación (False Accept Rate, FAR) representa, pues, la probabilidad de que acceda un individuo no autorizado y la probabilidad de falso rechazo (False Reject Rate, FRR) inciden en la frecuencia en que los usuarios legales son rechazados y, por tanto, han de repetir el intento de identificación. La FAR debe ser suficientemente baja, en un rango que suele establecerse entre 0.0001% y el 0.1%. Por ejemplo, en el 60% de las centrales nucleares de EE.UU. se emplean lectores de geometría de la mano con una FAR de 0.1 %. Hay que tener en cuenta que la tasa real de entradas no autorizadas resulta del producto de la FAR por la probabilidad de que un sujeto no autorizado alcance el dispositivo de control e intente el acceso. Si el sistema esta complementado con un elemento físico como una tarjeta magnética o un código numérico, por ejemplo, el intruso debe además poseer la tarjeta correspondiente ó una copia de la misma, o bien conocer el código de acceso.

La FRR debe también mantenerse baja para evitar el descontento de los usuarios y la ineficiencia del sistema. Por ejemplo, en un recurso con 1000 accesos diarios y una FRR del 1 % se producirían 10 incidencias diarias.

La validación de las tasas proporcionadas por los fabricantes no es fácil a causa de los porcentajes tan bajos que se manejan, exigiendo el examen supervisado de miles de accesos para obtener resultados significativos estadísticamente.

A continuación se describen los parámetros biométricos más utilizados, sus ventajas e inconvenientes.

*2.2.3.1 Huellas Dactilares*

La identificación por huellas dactilares es la mas antigua de las técnicas biométricas útiles y empleadas ampliamente. La huella dactilar resulta de la impresión dejada por los dedos en el papel mediante tinta, o en otro material por los propios fluidos exudados por la piel (huella latente), o bien de la exploración por parte de un dispositivo electrónico de las crestas papilares presentes en las yemas de los dedos.

Estas crestas configuran un patrón complejo que se considera único para cada individuo (en gemelos idénticos o univitelinos los patrones son similares, pero distinguibles). Existe evidencia científica de la extrema improbabilidad de que dos huellas dactilares procedentes de dos dedos distintos (del mismo o diferente individuo) coincidan por azar.

Tradicionalmente, las características extraídas de las huellas han sido, por un lado su tipo (clasificándose en varios tipos y subtipos de acuerdo a diversos esquemas y taxonomías, para facilitar su búsqueda, y por otro las minucias, que son bifurcaciones y finales abruptos de las crestas cuyas posiciones relativas identifican unívocamente la huella junto con la posición del centro y de unas estructuras denominadas deltas.

En una huella típica encontramos entre 50 y 100 minucias. Para obtener estas minucias se realiza un preproceso de la huella que filtra la imagen original, binariza y adelgaza las crestas evitando en lo posible la influencia de manchas, pequeñas cicatrices y residuos presentes en el momento de la impresión digital.

Además de comparar las minucias, existen otros procedimientos de comparación automática entre huellas basados en la correlación de las

imágenes de las crestas ya preprocesadas o de las direcciones de las mismas detectadas mediante filtros. Pese a que estos procedimientos poseen la potencialidad de conseguir excelentes resultados, presentan importantes dificultades debido a las deformaciones elásticas que sufren diferentes impresiones de un mismo dedo. Esto los hace en general poco eficientes para búsquedas en grandes conjuntos de huellas.

Ventajas:

- Alta universalidad. La ausencia de algún dedo o de una o ambas manos es relativamente poco frecuente.
- Alta permanencia. Se ha demostrado la invarianza esencial de las huellas dactilares a lo largo de toda la vida de un individuo.
- Alta unicidad. Existe abundante evidencia que demuestra la extrema improbabilidad de que huellas de dedos distintos sean idénticas.
- Buenas prestaciones. Existen algoritmos eficientes de comparación entre huellas. La información básica de las minucias puede almacenarse en poco espacio.
- Alta aceptabilidad. La larga tradición de uso de huellas dactilares genera una sensación de normalidad en la mayor parte de la población, aunque en casos esporádicos puede asociarse a criminalidad o invasión de la intimidad.

Desventajas:

- Media facilidad de medida. Los lectores electrónicos han llegado a tener costos muy bajos y son fáciles de instalar y mantener, aunque la adquisición de una buena impresión dactilar siempre se halla sujeta a la presencia de suciedad, cicatrices, heridas, etc. Así como muchos de los usuarios no saben colocar correctamente la huella en el lector.

- Aunque la aceptabilidad por parte de los usuarios es alta muchos de ellos resisten a tocar físicamente un sensor que ha sido utilizado previamente por mucha gente.

*2.2.3.2 Identificación por la voz*

La voz es uno de los rasgos que identificamos como particulares de las personas y, en la vida diaria, nos permiten reconocerlas con facilidad. Es un medio natural de interacción con el entorno y por tanto resulta muy aceptable para los usuarios pronunciar una palabra o frase ante un micrófono para identificarse.

Las características específicas de la voz de cada persona se deben a diferencias en aspectos fisiológicos y de comportamiento del aparato fonador. La forma del tracto vocal (laringe, faringe, cavidad oral, cavidad nasal, etc.) goza del papel más importante porque modifica fuertemente el contenido espectral de la onda sonora generada. Son precisamente las características del espectrograma de la voz las que configuran los parámetros biométricos usados habitualmente para distinguir un locutor de otro.

La gran variabilidad de la voz de un mismo individuo a lo largo de periodos relativamente cortos de tiempo, y la moderada especificidad de los parámetros que se extraen de ella hacen que el reconocimiento del locutor sea una técnica de verificación que se usa únicamente en combinación con identificación por tarjeta inteligente, por código de acceso, etc.

Ventajas:

- Alta facilidad de medida. El costo del "hardware" necesario es mínimo y la adquisición muy sencilla y cómoda para el usuario.

- Alta universalidad. El sector de la población con dificultades en el habla es relativamente reducido.
- Buenas prestaciones. Actualmente, la verificación es posible con recursos de cómputo muy bajos y el volumen de información almacenado es perfectamente aceptable con los medios de almacenamiento actuales.
- Alta aceptabilidad. Casi ningún usuario muestra reticencia a pronunciar una palabra o frase para acceder a un recinto o servicio.

Desventajas:

- Baja permanencia. Los parámetros básicos de la voz pueden alterarse fácilmente debido a muchos factores en periodos de tiempo muy cortos.
- Baja unicidad. La capacidad de distinguir un usuario de otro es solo moderada, ya que un importante parecido de los parámetros vocales no es raro.
- Baja resistencia al engaño. Una simple grabación de calidad preemitiría el acceso a no ser que la frase a pronunciar sea, por ejemplo, variable, ó haya de ser la respuesta a una pregunta realizada por el sistema de forma aleatoria, etc.

### *2.2.3.3 Reconocimiento Facial*

El reconocimiento facial, es decir, a través de la imagen del rostro, es uno de los que mayor crecimiento, al menos en cuanto a inversión y expectativas, esta experimentando actualmente. Se trata de un problema complejo, pero de gran interés, ya que el ámbito de aplicación es muy amplio. Por otro lado, también despierta importantes suspicacias en la población, fundamentalmente en los sectores especialmente preocupados por

los posibles perjuicios causados por las nuevas tecnologías en contra de la intimidad y las libertades individuales.

Se trata de un área de investigación activa actualmente y por tanto no existe consenso amplio todavía respecto al mejor tipo de características y los procedimientos de comparación mas adecuados. En cualquier caso, se trata de almacenar información local (ojos, nariz, boca, etc.) y global (posición de cada rasgo en la cara) e integrarla en un modelo que facilite la identificación y, en su caso, la búsqueda eficiente.

El reconocimiento de la cara no es un problema simple. La cara de una misma persona puede parecer muy diferente en diversas imágenes. Considerando las caras como objetos inflexibles, éstas pueden aparecer en diversas localizaciones, escalas u orientaciones. Pero las caras no son objetos inflexibles: las expresiones, los gestos o el habla también afectan el aspecto de la cara. Otras importantes fuentes de variabilidad son la iluminación, objetos artificiales (gafas de sol, etc.) o naturales (bigote, barba, etc.) que aparecen en la cara, el envejecimiento, etc. Algunas fuentes de variabilidad son mostradas en la Figura 3 donde se desea enfatizar que la iluminación, la expresión facial y la posición del rostro son fuentes de variabilidad.

Figura 3: Variabilidad en los rostros humanos

Un sistema típico podría constar de dos fases. En la primera se trata de localizar la cara en la imagen, distinguiéndola del fondo. En la segunda se caracteriza la misma y se comparan sus parámetros con los almacenados.

De la flexibilidad de la primera fase depende el rango de aplicaciones del sistema y de la precisión de la segunda, las prestaciones del mismo.

La aplicabilidad del reconocimiento facial en este momento no alcanza a aplicaciones de búsqueda en grandes conjuntos de "sospechosos" o accesos de alta seguridad si no va acompañada de sistemas clásicos como tarjetas o códigos personales. Los últimos intentos de aplicación a la localización de terroristas, de los cuales el más conocido es el de la policía de Florida en el aeropuerto de Tampa, han supuesto notorios fracasos.

Ventajas:

- Alta facilidad de medida. El costo del "hardware" (cámaras) es bajo y la adquisición puede incluso pasar inadvertida al usuario.
- Alta universalidad. Cualquier rostro no oculto por vestimenta es susceptible de verificación.
- Buenas prestaciones. La verificación es posible con recursos de cómputo razonables y la búsqueda lo es para conjuntos almacenados de tamaño pequeño o mediano (en el rango de pocos miles de caras) y el volumen de información almacenado es fácil de acomodar con los medios actuales.
- Alta aceptabilidad. Los usuarios no ven interrumpido su flujo de acceso, trabajo, etc.

Desventajas:

- Baja permanencia. El aspecto facial puede cambiar muy rápidamente debido a la aparición de barba, corte de pelo, uso de gafas, etc.

- Baja unicidad. La capacidad de distinguir un usuario de otro es actualmente moderada.
- Baja resistencia al engaño. El uso de disfraces y accesorios como gafas, sombreros, pañuelos, maquillaje, tintes, e incluso cortes de pelo ó peinados concretos pueden confundir al sistema. Otras formas de fraude como máscaras o fotografías son posibles, pero su uso se dificulta gracias a las capacidades 3D o termo-gráficas añadidas a algunos sistemas recientes.

## 2.2.4 Comparación de Sistemas Biométricos y otras consideraciones.

### *2.2.4.1 Comparaciones*

| Elemento/Sistema | Ojo (Iris)/Ojo (Retina) | Reconocimiento Facial |
|---|---|---|
| **Fiabilidad** | Muy alta | Media |
| **Facilidad de uso** | Media | Alta |
| **Prevención de ataques** | Muy alta | Baja |
| **Aceptación** | Media | Alta |
| **Estabilidad** | Alta | Media |

*Tabla 1: Sistema Biométrico Iris vs. Reconocimiento Facial*

| Elemento/Sistema | Huellas dactilares | Geometría de la mano |
|---|---|---|
| **Fiabilidad** | Alta | Alta |
| **Facilidad de uso** | Media | Alta |
| **Prevención de ataques** | Alta | Alta |
| **Aceptación** | Media | Alta |
| **Estabilidad** | Alta | Media |

*Tabla 2: Sistema Biométrico Huellas dactilares vs. Geometría de la mano*

| Elemento/Sistema | Escritura y firma | Voz |
|---|---|---|
| **Fiabilidad** | Alta | Alta |
| **Facilidad de uso** | Alta | Alta |

| | | |
|---|---|---|
| **Prevención de ataques** | Media | Media |
| **Aceptación** | Muy alta | Alta |
| **Estabilidad** | Media | Media |

*Tabla 3: Sistema Biométrico Escritura y firma vs. Reconocimiento voz*

A continuación se presenta los rasgos positivos y negativos propios de cada tecnología biométrica:

Rasgos positivos:

- Huellas dactilares: seguro y disponible especialmente para identificación. No acepta ni aún una cinta donde se haya levantado una impresión no visible a partir de una huella espolvoreada.
- Reconocimiento de Rostros: Apto para aplicaciones de identificación de uno contra muchos.
- Geometría de las manos: Fácil de usar.
- Escaneado de iris: Muy seguro para aplicaciones de identificación de uno contra muchos.
- Escaneado de retina: Muy seguro para aplicaciones de identificación.
- Análisis de la voz: Para aplicaciones de verificación local o remota siendo de bajo costo y no intrusivo.
- Verificación de Firma: Alto nivel de aceptación para verificación de un usuario determinado.

Rasgos negativos:

- Huellas dactilares: Resistencia al uso por connotaciones criminales.
- Reconocimiento de Caras: Costoso y sujeto a engaños con fotos montadas, específicamente sobre narices semejantes.

- Geometría de las manos: Sujeta a cambios físicos, no muy adecuada para grandes bases de datos de sistemas de identificación y verificación.
- Escaneado de iris: Costoso, sensible a los movimientos del usuario y ocupa mucho espacio en almacenamiento.
- Escaneado de retina: Costoso, no puede usarse con algunos usuarios por su sensibilidad a un escaneado infrarrojo o láser en los ojos.
- Análisis de la voz: Sujeto a cambios físicos y cierta facilidad de engaño con voces semejantes incluso con grabaciones en algunos casos.
- Verificación de Firma: Sujeta a cambios físicos.

En general en los sistemas biométricos lo que debe prevalecer o perseguirse para incrementar sus puntos a favor y disminuir los rasgos negativos son las siguientes características: fidelidad, exactitud y precisión.

Es necesario observar que en un sistema biométrico, normalmente al disminuir la tasa o porcentaje de aceptaciones equivocadas, también se incrementa la tasa de rechazos equivocados. Aunque la relación no sea necesariamente inversamente proporcional, la relación inversa es real.

En los sistemas biométricos también existe el fenómeno de que un aumento de la sensibilidad del sistema acarrea mayor procesamiento y retardo en el reconocimiento.

### *2.2.4.2 Costos de Implementación*

El costo de implementación de un sistema biométrico es una función que depende proporcionalmente de los siguientes factores:

- El hardware de captura biométrico.
- El mantenimiento de la base de datos
- Investigación y prueba del sistema biométrico
- Instalación, incluyendo sueldos del equipo que lo implementa
- Montaje, instalación, conexión, y los costes de la integración con los sistemas de usuario
- La educación del usuario, a menudo condicionada por campañas de marketing
- Pérdidas de la productividad debido a la curva de aprendizaje de la implementación
- Mantenimiento del sistema.

Actualmente el modelo de implementación de un sistema biométrico sugerido por las compañías dedicadas a este rubro consiste principalmente en la existencia de 2 servidores:

- Servidor de comparaciones: Equipo con altas prestaciones de memoria y procesamiento, capaz de ejecutar complejos algoritmos del núcleo del Sistema Biométrico.
- Servidor de Base de Datos y Conectividad: Equipo con altas prestaciones de memoria, procesamiento y almacenamiento, el cual contiene las características biométricas transformadas según el modelo matemático utilizado para representar dicha información.

Un servidor de altas prestaciones que cumple requisitos derivados de espacio, procesamiento y memoria requerido por los algoritmos es el HP COMPAQ DI385 G1 2400 DC cuyo costo es aproximadamente de: $13,373.55 (incluye impuestos)

| Cantidad | Descripción |
|---|---|
| 1 | HP DL380 G4 X3.4-2MB SCSI 800MHZ Front Side Bus |
| 1 | HP O280 2.4/10002M DC DL385G1 Kit (Procesador Dual-Core AMD Optaron 280 2.4 GHZ-1MB) |
| 2 | HP 2x2GB REG PC2-3200 Memory (4GB de Memoria RAM) |
| 5 | HP 5x300GB 10K U320 Pluggable Hard Drive (RAID5 de 300GB) |
| 1 | Fuente de Poder Redundante |
| 1 | Ventilador Redundante |

*Tabla 4: Costos de servidor apto para almacenamiento y proceso de algoritmos biométricos*

Adicionalmente a los servidores de aplicación y base de datos; probablemente sea necesario la adquisición de hardware adicional como escáner de rostros en 3D, o cámaras de vigilancia de alta resolución; estos costos son variables, debido a amplia existencia de hardware con algoritmos propietarios implementados internamente en los equipos, creados específicamente para tomar imágenes de rostro en 2D-3D a partir del video en tiempo real.

### *2.2.4.3 Aceptación por parte de usuarios*

La aceptación de los usuarios al empleo de la tecnología biométrica es inversamente proporcional al nivel de intrusión de la misma; sin embargo, pueden existir otros factores que ocasionen resistencia en mayor o menor grado, por ejemplo ciertos grupos de usuarios (grupos religiosos y grupos que defienden las libertades civiles) han rechazado estas tecnologías biométricas debido a preocupaciones relacionadas con la privacidad; también existe resistencia al uso de equipos biométricos relacionados a aspectos de seguridad e higiene: muchos usuarios no están de acuerdo en el uso de

equipos biométricos que implican el tacto en recintos donde existen mucha cantidad de infecciones en el medio (por ejemplo los nosocomios[7]); existen a su vez muchos mitos que deben ir desapareciendo sobretodo con la tecnología biométrica que interactúa con órganos muy sensibles del cuerpo humano, por ejemplo el escaneo del iris.

### *2.2.4.4 Biometría y su utilización según niveles de seguridad*

Las organizaciones deben determinar el nivel de seguridad que necesitan para sus aplicaciones específicas: bajo, moderado, o alto. Esta decisión afectará enormemente en la decisión de que tecnología biométrica es la más apropiada.

Generalmente, las técnicas biométricas relacionadas con el comportamiento son suficientes para las aplicaciones que requieren una seguridad baja o moderada; las técnicas biométricas físicas, son adecuadas para las aplicaciones con grandes requisitos de seguridad.

### *2.2.4.5 La estabilidad de la tecnología biométrica a largo plazo*

Las organizaciones deben considerar la estabilidad de la técnica biométrica, incluyendo la madurez de la tecnología, el grado de estandarización, el nivel de los fabricantes y las ayudas estatales, la cuota de mercado, y otros factores de soporte. Las tecnologías maduras y estandarizadas tienen generalmente una estabilidad más fuerte.

---

[7] Definición de Hospitales, donde debido a la naturaleza del ambiente son necesarias estrictas políticas de seguridad e higiene para evitar la contaminación por virus y bacterias.

## *2.3 Estándares asociados a tecnologías biométricas.*

En los últimos años se ha notado una preocupación creciente por las organizaciones reguladoras respecto a elaborar estándares relativos al uso de técnicas biométricas en el ambiente informático. Esta preocupación es reflejo del creciente interés industrial por este ámbito tecnológico, y a los múltiples beneficios que su uso aporta. No obstante ello, aún la estandarización continua siendo deficiente y como resultado de ello, los proveedores de soluciones biométricas continúan suministrando interfaces de software propietarios para sus productos, lo que dificulta a las empresas el cambio de producto o vendedor.

A nivel mundial el principal organismo que coordina las actividades de estandarización biométrica es el Sub-Comité 17 (SC17) del Joint Technical Committee on Information Technology (ISO/IEC JTC1), del International Organization for Standardization (ISO) y el International Electrotechnical Commission (IEC).

En Estados Unidos desempeñan un papel similar el Comité Técnico M1 del INCITS (InterNational Committee for Information Technology Standards), el National Institute of Standards and Technology (NIST) y el American National Standards Institute (ANSI).

Existen además otros organismos no gubernamentales impulsando iniciativas en materias biométricas tales como: Biometrics Consortium, International Biometrics Groups y BioAPI (este último compuesto por las empresas Bioscrypt, Compaq, Iridiam, Infineon, NIST, Saflink y Unisis).

*Los estándares más importantes son los siguientes:*

Estándar ANSI X.9.84: Estándar creado en el año 2001, por la ANSI (American National Standards Institute) y actualizado el 2003, define las condiciones de los sistemas biométricos para la industria de servicios financieros haciendo referencia a la transmisión y almacenamiento seguro de información biométrica, y a la seguridad del hardware asociado.

Estándar ANSI / INCITS 358: Estándar creado el año 2002 por ANSI y BioApi Consortium, que presenta una interfaz de programación de aplicación que garantiza que los productos y sistemas que cumplen este estándar son ínter operables entre sí.

Estándar NISTIR 6529: También conocido como CBEFF (Common Biometric Exchange File Format) es un estándar creado en 1999 por NIST y Biometrics Consortium que propone un formato estandarizado (estructura lógica de archivos de datos) para el intercambio de información biométrica.

## *2.4 Aplicación de Sistemas Biométricos en El Salvador*

El Salvador no es la excepción en la adopción de tecnología biométrica para implementar controles de acceso; a continuación se listan nombres de entidades jurídicas, control biométrico implementado y objetivo del control biométrico:

| Entidad | Sistema biométrico utilizado | Identificación adicional | Utilización |
|---|---|---|---|
| Hospital Nacional de Niños Benjamín Bloom | Geometría de la mano | Tarjeta física | Marcación y control de tiempo de asistencia |

| Entidad | Sistema biométrico utilizado | Identificación adicional | Utilización |
|---|---|---|---|
| Corte Suprema de Justicia | Geometría de la mano | Tarjeta física | Marcación y control de tiempo de asistencia |
| Centro de Datos, Aval Card S.A. de C.V. | Huella dactilar | Código PIN | Control de acceso a Centro de Datos |
| Registro Nacional de Personas Naturales | Huella dactilar | Tarjeta física | AFIS de registro civil de especificación mono-dactilar |
| Policía Nacional Civil | Identificación de Rostros | Ninguna | Identificación de sospechosos a través de retratos hablados o fotografías |
| Policía Nacional Civil | Huella dactilar | Ninguna | AFIS de Registro Criminal de especificación deca-dactilar |
| Banco Cuscatlán | Huella dactilar | Tarjeta física | Control de Acceso a Centro de Datos. |

*Tabla 5: Aplicaciones biométricas en El Salvador*

## *2.5 Hoja de Consultoría de Empresas Locales y Extranjeras.*

El mercado de tecnología Biométrica es dinámico, innovador y atractivo; por tal razón existen muchas iniciativas de investigación/implementación por parte de universidades y empresas privadas; entre las empresas de renombre por la experiencia en consultoría e implementaciones se encuentran las siguientes:

| Empresa | País de Origen | Productos Biométricos | Contactos |
|---|---|---|---|
| BlueStar Latin America. 2255 NW 102nd Place Miami, FL 33172. Phone: 305-470-6208 Fax: 305-470-6209 | Estados Unidos | WASP, B100 Biometric Clock w/ Wasptime Software / Huella dactilar combinada | Ing. Jorge Salinas Ing. Jorge Román www.bluestarinc.com sarias@bluestarinc.com |
| Vanguard Systems Internacional. 4111 Coral Tree Circle Suite 227 Coconut Creek, FL 33073. Phone: 954-990-9533 Fax: 561-892-3220 | Estados Unidos | iBASP/Intelligent Biometric Authentication Service Point / Huella dactilar combinada | Ing. Guillermo Gomez ggomez@vanguard-sys.com |
| Ex-Clé S.A. Soluciones Biométricas 225 Paraguay 1896 5° Buenos Aires C1121ABB, Argentina +54 (11) 4815.3488 | Argentina | Ex-Clé Finger Attendance 2.1 / Huella dactilar Ex-Clé Fingerprint Recover / Huella dactilar Ex-Clé Finger Guard / Huella dactilar - Reconocimiento Facial Ex-Clé Real-ID / Huella Dactilar | Ing. Hernan Sorell www.ex-cle.com hernan.sorell@ex-cle.com |
| Screen Check Docusal Edificio D'Cora | Holanda con representación en El Salvador | AFIS NEC / Huella dactilar combinada / AFIS Print Track NEC | Licda. Patricia García Lic. Faucy Brand Tel: (503) 2289-9909 Ext. 15 |

| Empresa | País de Origen | Productos Biométricos | Contactos |
|---|---|---|---|
| Urb. Santa Elena, Boulevard Orden de Malta Sur, Antiguo Cuscatlán, La Libertad, El Salvador, C.A. | | / Huella dactilar combinada | Fax: (503) 2289-9910<br>garcias@screencheck.com |
| Data Security Solutions S.A. San Francisco, Calle 50 Final, Edificio No. 14, Planta Baja, Panamá 69558<br>Phone: 507-226-1125<br>Fax: 226-1125 | Panamá | AFIS NEC / Huella dactilar combinada | Rogelio Morrel<br>Enrique Morgan<br>www.datasecuritys.com<br>rogelioc@datasecuritys.com |
| COASIN Costa Rica 50 Mts. Este de Of Central Pizza Hutt. Pavas. San José 12618-1000 Costa Rica<br>Phone: 506-296-8600<br>Fax: 506-296-7914 | Costa Rica | iBASP / Intelligent Biometric Authentication Service Point / Huella dactilar combinada | Kenneth Sanchez<br>www.coasin.com<br>amairena@coasin.co.cr |
| Informática Integrada 11ª avenida 32-35 Zona 5 Guatemala 1005 | Guatemala | BIO-KEY / WEB-KEY Solution / Huella dactilar combinada | Carlos Chian<br>chrcorp@terra.com.gt |

| Empresa | País de Origen | Productos Biométricos | Contactos |
|---|---|---|---|
| Phone: 502-362-0680<br>Fax: 502-331-9989 | | | |
| Redecon<br>19 Avenida B 0-83 Zona 15, Vista Hermosa 2, Guatemala, 1015 | Guatemala | iBASP / Intelligent Biometric Authentication Service Point / Huella dactilar combinada | Lic. Danilo Herrera<br>Phone: 502-2385-7219<br>Fax: 502-2369-6592<br>www.redecon-sa.com |
| Sistemas C&C<br>Avenida Olímpica No. 3222, San Salvador, El Salvador | El Salvador | AFIS NEC / Huella dactilar combinada / AFIS Print Track NEC / Huella dactilar combinada | Ing. Annie Orellana de Tona<br>Phone: (503) 2298-4777<br>www.sistemascc.com<br>arrellana@sistemascc.com |

*Tabla 6: Experiencia con Consultoría de Empresas Locales y Extranjeras*

# 3. OBJETIVOS

## *3.1 Objetivo General*

- Investigar y desarrollar un prototipo de Sistema Biométrico para la identificación de rostros humanos utilizando la teoría de Eigenvectores y Eigenvalores.

## *3.2 Objetivos Específicos*

- Realizar una investigación teórica sobre el área del algebra lineal conocida como Eigenvectores y Eigenvalores, destacando su utilidad en el área de la ingeniería y su relación con el reconocimiento de imágenes en el área de la visión artificial[8].

- Investigar los principales algoritmos para el tratamiento de imágenes digitales y señales en general; generando comparaciones entre algoritmos y destacando su utilidad y formas de implementación en ambientes OpenSource, los cuales no implican alza en costos y se encuentran disponibles para diversos sistemas operativos.

- Implementar algoritmos de pre-tratamiento de imágenes para el mejoramiento de las imágenes faciales; dichos algoritmos pretenden

[8] La Visión artificial, también conocida como Visión por Computador (del inglés Computer Vision) o Visión técnica, es un subcampo de la inteligencia artificial. El propósito de la visión artificial es programar un computador para que "entienda" una escena o las características de una imagen.

ser reutilizados en el transcurso del proyecto, específicamente en el ámbito del tratamiento y procesamiento de imágenes.

- Aplicar la teoría de Eigenvectores y Eigenvalores para reconocer a través de subespacios, un rostro humano de un conjunto limitado de fotografías de rostros; de esta forma resaltar la importancia del área de las matemáticas como mecanismos de solución a problemas reales y complejos relacionados con visión artificial.

- Desarrollar un prototipo que permita el reconocimiento de rostros faciales enfocados; la base del desarrollo del prototipo será un trabajo de investigación que facilite el desarrollo de proyectos futuros de reconocimiento facial y visión artificial que sean desarrollados por la Universidad Don Bosco.

# 4. ALCANCES Y LIMITACIONES

## *4.1 Alcances*

- Se pretende desarrollar un trabajo de investigación que documente de forma didáctica los algoritmos para el procesamiento de imágenes a utilizar; estos algoritmos abren muchas oportunidades para diseñar aplicaciones relacionadas con el reconocimiento de información en imágenes.

- Implementar aquellos algoritmos para el procesamiento de imágenes, que resulten útiles en la solución de problemas relacionados con el pre-tratamiento y post-tratamiento de imágenes.

- Implementar las teorías sobre eigenvectores y eigenvalores, enfocados en el método PCA (Principal Component Analysis) para obtener como resultado final un prototipo para la identificación de Rostros Humanos.

- La implementación de los algoritmos se hará utilizando herramientas que garanticen portabilidad y que su utilización no este condicionada por costos de licenciamiento, los cuales en algunos casos limitan el desarrollo o continuación de prototipos y proyectos.

## *4.2 Limitaciones*

- El prototipo se diseñará para trabajar con imágenes de rostros faciales bien identificados, es decir fotografías similares a las que se utilizan en documentos legales, registros civiles o policiales.

- El prototipo trabajará con imágenes de un tamaño predeterminado, no sobrepasando los 256x256 píxeles, las imágenes pueden estar en los formatos JPG, TIFF, BMP y PNG.

- Las fotografías a utilizar deben estar correctamente alineadas, y el rostro de las personas debe encontrarse centrado en el espacio de la fotografía; aunque existen algoritmos específicos a la biométrica, que permiten la identificación mas precisa del rostro humano, estos no serán explorados en este trabajo; sin embargo puede ser abordado en otro trabajo posterior.

- El prototipo a desarrollar utilizará los algoritmos contenidos dentro de la teoría PCA, y otros algoritmos para el tratamiento de imágenes.

- No se implementarán en el prototipo, algoritmos de lógica difusa y de redes neuronales especializados en el reconocimiento biométrico de rostros, específicamente aquellos utilizados para la identificación de particularidades tales como: los ojos, la nariz, la boca, la barbilla; Determinando la posición, tamaño, longitudes y distancias de cada una de estas características en el rostro.

# 5. PLANTEAMIENTO Y JUSTIFICACION DEL PROYECTO

La tecnología biométrica actualmente está siendo ampliamente reconocida, evaluada e incorporada en los sistemas de uso cotidiano. Las razones que motivan el desarrollo e implementación de la biometría son varias, entre las cuales se pueden mencionar:

- Necesidad de seguridad contra terrorismo y actos de violencia en zonas amplias de difícil monitoreo como aeropuertos, centros comerciales y otros.
- Validación de identidad en trámites muy importantes realizados en entidades de migración y sectores financieros.
- El avance de la tecnología informática en cuanto a hardware/software permite implementaciones ajustables a la relación costo/beneficio que buscan las empresas e instituciones.

Actualmente grandes empresas dedicadas al sector electrónico/informático están impulsando el desarrollo de estas tecnologías, las cuales continuamente están evolucionando para desarrollar mejores algoritmos, mayor exactitud en los resultados, menores costos de implementación, mayor flexibilidad y transparencia para los usuarios, etc. Se puede afirmar que la biometría aunque sus antecedentes se remontan a finales del siglo XIX no es una ciencia terminada; por el contrario es una ciencia que siempre se enfrenta a retos complejos, relacionados con el campo de la visión artificial, que no han sido superados completamente, a pesar del avance tecnológico e informático actual reflejado en la capacidad, velocidad y tamaño de dispositivos electrónicos.

El campo de la biometría junto con el de la visión artificial son áreas de investigación donde deben enfocarse esfuerzos para el desarrollo de mecanismos transparentes y eficientes que sean aplicables de forma flexible a la vida cotidiana de información en la que se vive inmerso.

# 6. TÉCNICAS EXISTENTES PARA RECONOCIMIENTO FACIAL

## *6.1 Introducción*

Los modelos computacionales para el reconocimiento de rostro son interesantes debido al aporte que hacen en el campo de la teoría como el gran número de aplicaciones practicas que permitiría desarrollar que abordan temas complicados y de suma importancia como lo son identificación de criminales, sistemas de seguridad, procesamiento de imágenes/videos y la interacción humano-computadora.

Desafortunadamente, el desarrollo de un modelo computacional para el reconocimiento de rostro es complicado debido a lo complejo, multidimensional y cantidad de información significativa presente en un rostro lo cual implica definición de tareas que simulen la actividad neuronal humana.

Existen numerosas aproximaciones desarrolladas por la ciencia para lograr la tarea de identificar rostros humanos; a continuación se enumeran los principales.

- Principal Component Analysis (PCA)
- Independent Component Analysis (ICA)
- Linear Discriminant Analysis (LDA)
- Evolutionary Pursuit (EP)
- Elastic Bunch Graph Matching (EBGM)
- Kernel Methods
- Trace Transform

- Active Appearance Model (AAM)
- Morphable Model
- 3-D FACE Recognition
- Bayesian Framework
- Support Vector Machina (SVM)
- Hidden Markov Models (HMM)
- Boosting & Esenmble

En el presente proyecto se abordará la técnica PCA; el cual es una técnica que aborda la imagen facial en un problema de 2 dimensiones utilizando un conjunto de vectores conocidos como Eigenvectores.

## *6.2 Métodos Existentes para la identificación facial*

### 6.2.1 Principal Component Analysis (PCA)

Un problema central en el análisis de datos multivariantes es la reducción de la dimensionalidad: si es posible describir con precisión los valores de $p$ variables por un pequeño subconjunto $r < p$ de ellas, se habría reducido la dimensión del problema a costa de una pequeña pérdida de información.

El análisis de componentes principales tiene este objetivo: dadas $n$ observaciones de $p$ variables, se analiza si es posible representar adecuadamente esta información con un número menor de variables construidas como combinaciones lineales de las originales. Por ejemplo, con variables con alta dependencia es frecuente que un pequeño número de nuevas variables (menos del 20 por 100 de las originales) expliquen la mayor parte (más del 80 por 100 de la variabilidad original).

La técnica de componentes principales es debida a Hotelling (1933), aunque sus orígenes se encuentran en los ajustes ortogonales por mínimos cuadrados introducidos por K. Pearson (1901). Su utilidad es doble:

1. Permite representar óptimamente en un espacio de dimensión pequeña observaciones de un espacio general p-dimensional. En este sentido, componentes principales es el primer paso para identificar las posibles variables latentes, o no observadas que generan los datos.
2. Permite transformar las variables originales, en general correladas, en nuevas variables incorreladas, facilitando la interpretación de los datos.

### 6.2.2 Independent Component Analysis (ICA)

El análisis de componentes independientes, en adelante ICA, fue presentado en 1986 por Jeanny Herault y Christian Jutten en Utah como una red neuronal basada en la ley de aprendizaje de Hebb capaz de realizar una separación ciega de señales. En concreto, este algoritmo trata de separar un número determinado de señales estadísticamente independientes a partir un número idéntico de señales de entrada que son suma lineal de las primeras.

La primera aplicación inmediata de ICA, es la eliminación de ruido; se trata de separar estas últimas señales no deseadas pudiendo realizar la clasificación únicamente sobre las señales originales resultado de la actividad neuronal.

La restricción en el uso de esta técnica radica en sus condiciones de aplicación, éstas, para un caso general, son:

- Las fuentes, es decir, las señales originales que se mezclan y que ICA deberá recuperar posteriormente, deben ser linealmente independientes
- El retardo de propagación a través del medio en el que se mezclan las señales tiene que ser despreciable
- Las señales originales deben ser analógicas y su función de distribución de probabilidad no puede ser gaussiana.
- El número de componentes independientes es el mismo que el de señales originales.

### 6.2.3 Linear Discriminant Analysis (LDA)

Las herramientas estadísticas tales como las técnicas discriminantes permiten desarrollar habilidades de manejo de información para la investigación y solución de diversos tipos de problemas. El análisis discriminante es una técnica multivariable que permite llevar a cabo tareas que dan pie para tomar ciertas decisiones en situaciones reales, tales como:

1. Distinguir entre diversos grupos mutuamente excluyentes. Como puede ser entre buenos y malos clientes de una empresa; alumnos responsables e irresponsables en una institución, distinguir o clasificar las observaciones de una investigación, detectar el por qué de las diferencias, pronosticar a qué grupo pertenecerá una persona de acuerdo a sus características, entre otros.
2. Identificar las variables que son importantes para distinguir entre los grupos y con el fin de desarrollar un procedimiento para predecir la membresía de aquellos casos que no han sido estudiados. Como puede ser el caso de responder a una solicitud de préstamo de un cliente en una empresa, o la solicitud de empleo por parte de un estudiante de la institución.

La aproximación clásica de Fisher (1936) para LDA esta basada en la escogencia de combinaciones lineales de las variables para maximizar la varianza entre grupos y minimizar la varianza dentro de grupos, estas combinaciones lineales de las observaciones originales se denotan como puntajes:

$$z_i^{(1)} = \sum_{l=1}^{p} W_l X_{il}^{(1)}, i = 1,...,n_1 \qquad z_j^{(2)} = \sum_{l=1}^{p} W_l X_{jl}^{(2)}, j = 1,...,n_2$$

Donde $X_{jl}^{(1)}$ es la observación $i$ de la variable $l$ del Grupo (Clase) 1 y $X_{jl}^{(2)}$ es la observación $j$ de la variable $l$ del Grupo (Clase) 2. La función discriminante esta determinada por el vector $W_p = (W_1,...,W_p)$.

El procedimiento de Fisher maximiza la varianza entre grupos y minimiza la varianza dentro de grupos, dada por la razón $\frac{S_B^2}{S_w^2}$, donde $S_B^2$ es la varianza entre grupos y $S_W^2$ es la varianza dentro de grupos. Esta aproximación es equivalente a la maximización de la razón de correlación de Pearson $\eta^2 = \frac{S_B^2}{(S_B^2 + S_W^2)}$.

La maximización ya mencionada constituye el procedimiento familiar de "separar" dos grupos tanto como sea posible. El estadístico $\eta^2$ puede escribirse como

$$\eta^2 = \frac{N\left(\overline{Z}^{(1)} - \overline{Z}^{(2)}\right)^2}{N\left(\overline{Z}^{(1)} - \overline{Z}^{(2)}\right)^2 + S^2}$$

Donde $\overline{Z}^{(1)}$ y $\overline{Z}^{(2)}$ son las medias aritméticas de los puntajes $Z_i^{(1)}$ y $Z_j^{(2)}$ $N = n_1 * n_2/(n_1 + n_2)$, $S^2 = \left[(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2\right]/(n_1 + n_2 - 2)$ y $S_1^2$ y $S_2^2$ denotan las varianzas muestrales de las dos distribuciones de puntajes, respectivamente. La razón de correlación $\eta^2$ varía entre 0 y 1. La bondad de "separación" entre los dos grupos de puntajes pueden ser medidos por la desviación de $\eta^2$ a 1, o bondad teórica superior.

La función discriminante lineal para LDA es el vector $W_p^{(l)}$ que maximiza la cantidad $\eta^2(W_p)$.

6.2.4 Evolutionary Pursuit (EP) - Método de Seguimiento Evolutivo.

La idea del Método de Seguimiento Evolutivo desarrollado por Liu y Wechsler fue comenzar con una reducción de dimensionalidad usando PCA. Luego se aplica una transformación lineal llamada Transformada Whithening, gracias a la cual se logra obtener una base no ortogonal en el sistema completo. En este nuevo espacio, WPCA (Whithened - PCA), se busca el mejor conjunto de vectores de proyección mediante rotaciones de pares de ejes y escogiendo un subconjunto de ellos, tal que maximicen la función objetivo. El número de variables de optimización dado por los ángulos de rotación y los ejes seleccionados es típicamente muy grande, y por lo tanto conviene usar técnicas de optimización especiales. Por esta razón Liu y Wechsler optaron por usar Seguimiento Evolutivo (EP: Evolutionary Pursuit), una forma especial de Algoritmos Genéticos (GAs), para optimizar la función objetivo. Al igual que todos los métodos lineales, los vectores de proyección obtenidos con este método pertenecen al espacio original de imágenes y son conocidos como EP-Faces.

### 6.2.5 Elastic Bunch Graph Matching (EBGM).

Elastic Graph Matching (EGM) recibe este nombre porque un grafo etiquetado es la estructura de alto nivel elegida para representar la cara. Esta estructura es compuesta por un conjunto de nodos, que, corresponden físicamente a la posición de algunas características faciales bien definidas (por ejemplo, la extremidad de la nariz) y de un sistema de arcos que conectan los nodos que pertenecen a las mismas características faciales (por ejemplo, la boca). La elasticidad se requiere para poder emparejar el gráfico a cualquier imagen de la cara.

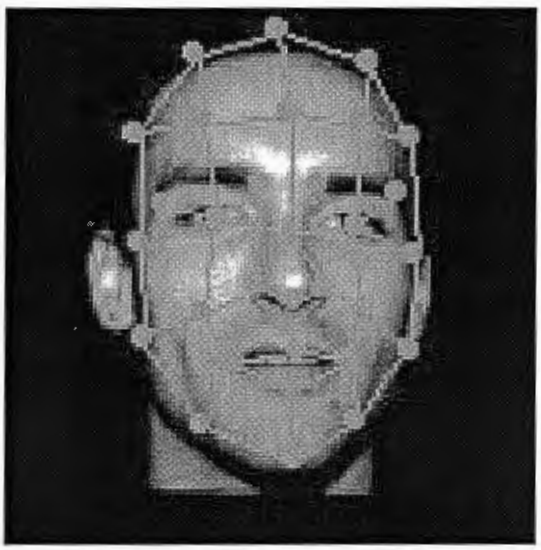


*Figura 4: Marcación de Características en el rostro utilizando EBGM*

Para incluir características de la textura en la descripción de cada nodo se atribuye un vector llamado "jet". La descripción consiste en las respuestas de un sistema de los filtros (filtros de Gabor) en el nodo dado. Se aplica una familia de los filtros de Gabor indexados por sus frecuencias centrales kj.

La siguiente figura muestra las imágenes filtradas para una imagen dada (imagen de arriba) y el banco de los filtros de Gabor. Tres escalas (filas) y 4 orientaciones (columnas) compusieron el banco de filtro.

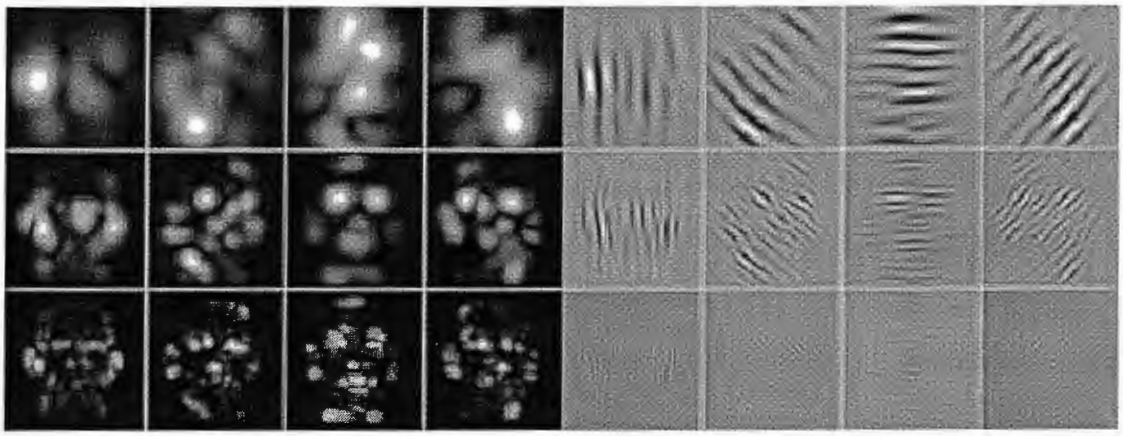


*Figura 5: Ejemplos de imágenes filtradas. El bloque de la izquierda muestra las respuestas en módulo y el de la derecha la parte real. En cada bloque, cada columna representa una orientación de análisis distinta y cada fila una escala diferente.*

Una estructura de grafo genérica (el Face Bunch Graph) que contiene, en cada nodo, todos los jets de un conjunto de entrenamiento dado, se empareja a una nueva imagen entrante. Una vez que se haya realizado el emparejamiento, el reconocimiento se hace por medio de la comparación del grafo obtenido y los grafos almacenados en una base de datos de entrenamiento. En ambas etapas de emparejamiento y de reconocimiento, una medida de la semejanza entre las dos estructuras de grafos es crucial.

### 6.2.6 Kernel Methods.

Los Kernel Methods o Métodos de Kernel en español, también conocidos como Estimadores Focales/Análisis de Patrones Puntuales se basan en la disposición de un conjunto de eventos sobre una región del plano y se enmarca en una de las tres grandes ramas de la estadística espacial, la de los procesos puntuales. Básicamente, pretende determinar si dichos eventos presentan un patrón de agregación (los eventos se producen cerca de otros

eventos), de inhibición (los eventos aparecen diseminados) o de aleatoriedad espacial completa (los eventos se producen con igual probabilidad en cualquier punto del espacio, con independencia de dónde se hallen otros eventos). Además es posible la comparación de los patrones de dos conjuntos de eventos y, si el patrón es de agregación o de inhibición, puede considerarse su modelización como proceso puntual, lo que puede permitir análisis estadísticos que generen mayor información.

Ya se han aplicado con éxito en algunas situaciones de interés en epidemiología, tales como el análisis de posibles fuentes de contaminación alrededor de las cuales surgen casos de alguna enfermedad. Su campo de aplicación es más amplio, y puede incluso servir también para cartografiar enfermedades.

Un patrón puntual está formado por un conjunto de *n* eventos que han ocurrido sobre unas determinadas coordenadas $\{(x_1, y_1),...,(x_n, y_n)\}$ del plano. A semejanza de la descripción que se realiza de cualquier conjunto de observaciones, mediante una medida de tendencia (media o mediana) y una de dispersión (desviación típica), un patrón puntual se puede describir mediante una medida de tendencia, como la densidad de eventos en la región, y mediante una medida del grado de dispersión o agregación de los mismos. Estas dos características se corresponden con dos funciones, a saber:

1. La función de intensidad $\lambda(x,y)$, que mide la densidad de casos por unidad de área en el punto $(x,y)$. La estimación de la función de intensidad en un punto cualquiera $(x,y)$ de la región en estudio se hace a partir de la posición $\{(x_1,y_1),...,(x_n,y_n)\}$ de los eventos, y puede realizarse de forma puntual o mediante una función auxiliar llamada «núcleo» ó «kernel» . Esta última forma es la más común,

ya que la primera puede considerarse un caso particular suyo. Su expresión es:

$$\hat{\lambda}_t(x,y) = \sum_{t=1}^{n} f_t[(x,y)-(x_t,y_t)]$$

*Figura 6: función intensidad (kernel method)*

2. La función *K*(*s*) de Ripley, que mide la agregación de los eventos a una cierta distancia *s* y que se define según:

$$K_{(s)} = \frac{promedio\ de\ eventos\ a\ dis\tan cia \le s\ de\ otro\ evento}{\lambda}$$

*Figura 7: función K(s) de Ripley*

Donde el numerador se estima mediante una función tipo núcleo también y el denominador se estima como la densidad media de eventos en la región bajo estudio.

6.2.7 Trace Transform.

Esta técnica es una generalización de la "Random Transform" o transformada aleatoria, la cual consiste en reconstruir una imagen a partir de líneas ejes las cuales pertenecen a ciertas funciones de la imagen. El método Trace Transform convierte la imagen original en otra imagen la cual depende de los parámetros $(\phi\rho)$ los cuales caracterizan cada una de las líneas; ver Figura 8.

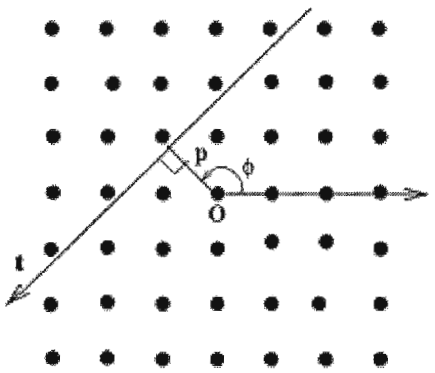


*Figura 8: Definición de líneas en Trace Transform*

En la Figura 8 el parámetro t es definido como la línea base u origen desde donde se generarán todas las líneas basadas en funciones.

Las propiedades principales de esta técnica son las siguientes:

- No es afectada por la rotación, traslación o modificación en escalas.
- Los parámetros de dos imágenes pueden ser recuperados independientemente de su rotación, traslación o modificación en escala.
- Son ideales para identificar una secuencia de imágenes.

Debido a las propiedades mencionadas anteriormente; esta técnica se utiliza principalmente en:

- Desarrollo de inventarios de imágenes en bases de datos, donde los objetos son almacenados para su posterior utilización en la identificación de imágenes sin importar rotación, traslación o escala.
- La identificación de los parámetros de la transformación de una imagen comparados contra los parámetros recuperados de otra imagen pueden ser utilizados para determinar si se trata de la misma imagen o diferentes; esto es útil para detectar fraudes.
- El tratamiento que permiten en imágenes secuenciales es útil para establecer sistemas de monitoreo; sistemas de detección de cambios y vigilancia en general.

Esta técnica parte del hecho que una imagen está sujeta a distorsiones lineales las cuales son rotación, modificación en escala y traslación. Se asume un sistema de coordenadas originales $C_1$ y un sistema de coordenadas distorsionadas $C_2$. $C_2$ es obtenido de $C_1$ por rotación de ángulos

(parámetro: $-\theta$) , por una modificación en la escala de los ejes (parámetro: $v$) y por la traslación con relacionada con el vector: $(-s_0 \cos\psi_0, -s_0 sen\psi_0)$.

De esta forma, una imagen puede ser mostrada bajo un nuevo sistema de coordenadas donde las nuevas líneas han sido generadas a partir de estas transformaciones lineales); si una línea en $C_2$ está parametrizada por $(\Phi, \rho, t)$ en el viejo sistema de coordenadas $C_1$ los parámetros son: $\Phi_{old} = \Phi - \theta$, $\rho_{old} = \upsilon[\rho - s_0 \cos(\psi_0 - \Phi)]$ y $t_{old} = \upsilon[t - s_0 sen(\psi_0 - \Phi)]]$.

Si se considera una imagen con todas las líneas generadas en todas las direcciones, se define $\Lambda$ a dicho conjunto que abarca todas estas lineas; de esta forma la técnica Trace Transform es una función $g$ definida en $\Lambda$, la cual se auxilia de una función $T$ la cual depende de la variable $t$ y es conocida como "trace functional"; de esta forma, si $L(C_1; \Phi, \rho, t)$ es una línea en el sistema de coordenadas $C_1$, entonces: $g(F; C_1; \Phi, \rho) = T(F(C_1; \Phi, \rho, t))$

## 6.2.8 Active Appearance Model (AAM).

La técnica AAM (Active Appearance Models) consiste en la construcción un modelo estadístico de forma y de textura de un objeto a partir de las propiedades geométricas y de la textura observada en un conjunto de imágenes de entrenamiento. Este conjunto contiene imágenes de objetos de una misma clase inicial (en nuestro caso los rostros) marcados con una serie de puntos denominados "landmarks". El modelo estadístico está compuesto por el contorno y textura medias del conjunto de entrenamiento, junto con las direcciones de variación respecto a esos valores medios.

Estas direcciones de variación se obtienen de la matriz de covarianza creada con las de las imágenes de entrenamiento marcadas. El modelo estadístico permite generar ejemplares válidos del objeto a ajustar sobre otro objeto nuevo que se encuentre en una nueva imagen. Es por tanto, una técnica top-down: primero se genera el ejemplar del modelo y después se deforma según las direcciones de variación para ajustarlo a la imagen.

El proceso completo de análisis de una imagen mediante AAM, está constituido por dos fases principales:

- Fase de construcción del modelo AAM (entrenamiento).
- Fase de ajuste del modelo AAM (reconocimiento).

La fase de construcción del modelo abarca diferentes tareas: la elección de las imágenes de entrenamiento, elección de puntos de control ("landmarks") adecuados, marcado de las imágenes, alineamiento y normalización del conjunto de entrenamiento, cálculo del modelo estadístico de forma y cálculo del modelo estadístico de textura. Esta etapa se realiza una sola vez. Una vez se ha obtenido el modelo, se puede utilizar tantas veces como sea necesario.

La fase de ajuste toma de punto de partida el modelo ya construido e intenta ajustarlo sobre el objeto de una imagen nueva. Para llevar a cabo esto, la forma o contorno medio de nuestro modelo será proyectado sobre la imagen. Mediante sucesivas iteraciones, se va deformando modificando los parámetros dentro de los límites establecidos, de manera que nunca deje de ser un ejemplo válido.

Cuando acaba este proceso de ajuste, el contorno resultante será una realización del modelo con unos parámetros únicos y que se ajustará lo más posible a la imagen origen. El uso de la técnica AAM, para realizar el proceso de ajuste sobre un objeto de una imagen nueva, requiere que el contorno del modelo se halle próximo al contorno del objeto de la imagen nueva. La

colocación del contorno del modelo, de la que partimos para realizar el proceso iterativo de ajuste al objeto de la imagen nueva, recibe el nombre de "inicialización del sistema AAM".

### 6.2.9 3D Morphable Model.

3D Morphable Model es una técnica utilizada para generar modelos de rostros en tres dimensiones; estos modelos se generan a partir de tres o más imágenes de cada persona las cuales se encuentran almacenadas en una base de datos de entrenamiento. Los modelos en 3D son renderizados a partir de la variación de poses y las condiciones de iluminación presente en las imágenes; el resultado de esta técnica en sistemas de reconocimiento facial a alcanzado un 90% de exactitud sobre una base de datos de 1,200 imágenes de 6 personas; adicionalmente esta técnica muestra un velocidad de reconocimiento muy aceptable.

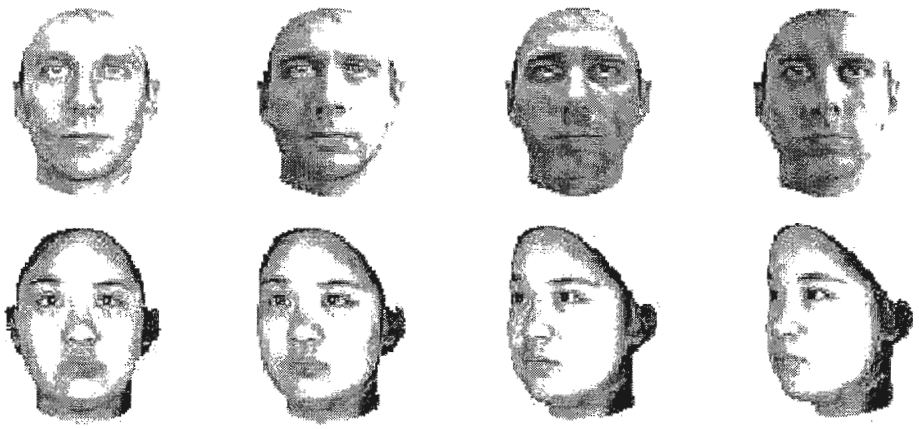


*Figura 9: Generación de modelos en 3D a partir de diferentes posiciones e iluminación de las imágenes.*

Esta técnica es implementada detectando de un rostro los componentes los cuales son utilizados después para reconocer el rostro. La arquitectura del sistema es dividida esquemáticamente en 4 pasos:

1. Identificar la sección del rostro (área del rostro excluyendo cabello, hombros, cuello)
2. Aplicar modelos Lineales SVM para identificar los ojos, la nariz y la boca.
3. Realizar un renderizado producto de los modelos lineales SVM recolectados de las imágenes de entrenamiento de una misma persona.

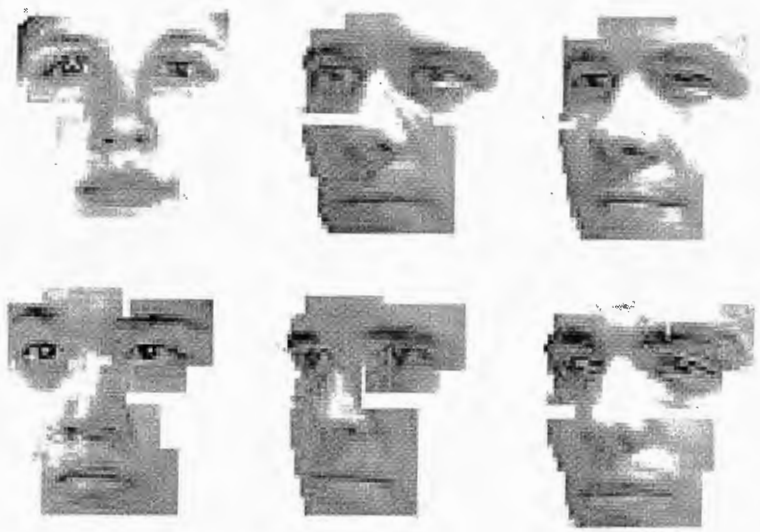


*Figura 10: Reconstrucción 3D de rostros basándose en partes del rostro obtenidos a partir de modelos lineales SVM*

6.2.10 3-D FACE Recognition.

El reconocimiento en 3D es una técnica basada en las características geométricas físicas individuales e invariables. La idea principal del algoritmo es la representación de la superficie facial del rostro, el cual tiene características invariantes y deformaciones isométricas producto de las expresiones faciales. Una de las etapas cruciales en la construcción de estas

medidas invariantes geométricas es la medida de distancias geodésicas en superficies triangularizadas, las cuales son ejecutadas en dominios triangularizados (FMTD).

El surgimiento de esta técnica nace debido a las limitantes que existen en el reconocimiento en 2D; donde las imágenes se ven afectadas por la iluminación y la posición, además se basan en la transformación Euclidea la cual compara 2 superficies y determina maximiza su similitud, sin embargo la transformación Euclidiana es válida para objetos rígidos; y el rostro humano no puede ser considerado como un objeto rígido debido a las múltiples deformaciones causadas por las expresiones faciales.

Las expresiones faciales pueden ser modeladas como transformaciones isométricas (o de larga duración); estas generan las superficies isométricas.

El núcleo de la teoría de construcción de superficies isométricas es conocido como: *bending-invariant canonical forms*; y parte de una aproximación poliédrica del rostro facial; donde se puede considerar que la superficie facial puede ser obtenida a partir de un conjunto finito de puntos $p_i(i=1,...,n)$ con una métrica discreta $\delta$ asociada a dicha superficie: $\delta(p_i,p_j)=\delta_{ij}$

La métrica discreta puede ser escrita en una forma matricial de distancias mutuas entre los puntos de la superficie; por conveniencia se definen distancias mutuas cuadradas: $(\Delta)_{ij}=\delta_{ij}^2$

La matriz $\Delta$ es invariante en términos de deformaciones de la superficie isométrica; pero no constituye en si misma una única representación de la superficie isométrica debido a su dependencia en el orden de los puntos

$p_i (i = 1,...,n)$; el objetivo de esta técnica es encontrar una representación geométrica invariable única.

Si se tratan las distancias cuadradas mutuas como un caso particular de disimilitudes se puede utilizar una técnica de reducción dimensional llamada *multidimensional scaling (MDS)* para reducir el espacio dimensional en un espacio Euclidiano $R^M$ : $\varphi : (S, \delta) \to (R^m, d), \varphi(\rho_u) = x_i$ y minimizar el error: $\varepsilon = f(|\delta_{ij} - d_{ij}|), d_{ij} = \|x_i - x_j\|_2$

El paso siguiente, después de encontrar una representación geométrica isométrica única es realizar un algoritmo para el cálculo de las distancias geodésicas en la superficie $\delta_{ij}$; el algoritmo sugerido para tal tarea es el FMTD (fase marching on triangulated domains) el cual tiene una complejidad de $O_n$

6.2.11 Bayesian Framework.

El método Bayesiano (Bayesian Framework) sugiere un esquema de segmentación y otro de clasificación. La segmentación divide a una imagen en un conjunto de regiones.

Para clasificar estas regiones, se divide el conjunto de píxeles que componen a la imagen en clases temáticas previamente definidas. A continuación se presenta la segmentación y clasificación de las regiones utilizando la función discriminante de Bayes, la cual está dada por la siguiente expresión:

$$p(\omega_k | x) = \frac{p(\omega_k) p(x | \omega_k)}{p(x)}$$

donde:

$p(x) = \sum_{k=0}^{M-1} p(\omega_k) p(x|\omega_k)$, es la probabilidad total del evento/píxel x;

$p(\omega_k)$, es la probabilidad a priori de la ocurrencia de la clase: $\omega_k$;

$p(x|\omega_k)$, es la probabilidad condicional del píxel $x$, dada la clase $\omega_k$;

$M$, es el número de clases o regiones típicas.

La expresión de probabilidad a posteriori señala la probabilidad de que ocurra un evento $W_x$ dado que x ya ocurrió y que en una aproximación tipo máximo a posteriori constituye la etapa final del cálculo. Para aplicar la expresión anterior, hemos definido el proceso de segmentación en dos partes: la primera es la pre-clasificación bayesiana y la segunda es la clasificación.

**Preclasificación Bayesiana**

Se define una etapa de preclasificación en virtud de desconocer las probabilidades a priori $p(\omega_k)$. Esta etapa consta de dos partes de procesamiento.

Primera parte

1. Sobre la imagen de prueba se definen 3 ventanas de entrenamiento que identifican a tres clases típicas: la clase 1 con tonos de gris muy blancos, la clase 2 opaca y la clase 3 con tonos oscuros.
2. Sobre cada ventana se calcula la media $\overline{X}_k$ de la clase $\omega_k$.
3. Se aproximan las probabilidades conjuntas del numerador de la regla de Bayes mediante las funciones mostradas en la figura a continuación, las cuales indican la probabilidad condicional de que un píxel dado pertenezca a la clase $\omega_k$. Tales funciones identifican a cada

clase y corresponden en el histograma a los datos en nivel de gris de la imagen a procesar. Como parámetros se emplean las medias y medidas de dispersión visibles en los traslapes de las funciones adjuntas.

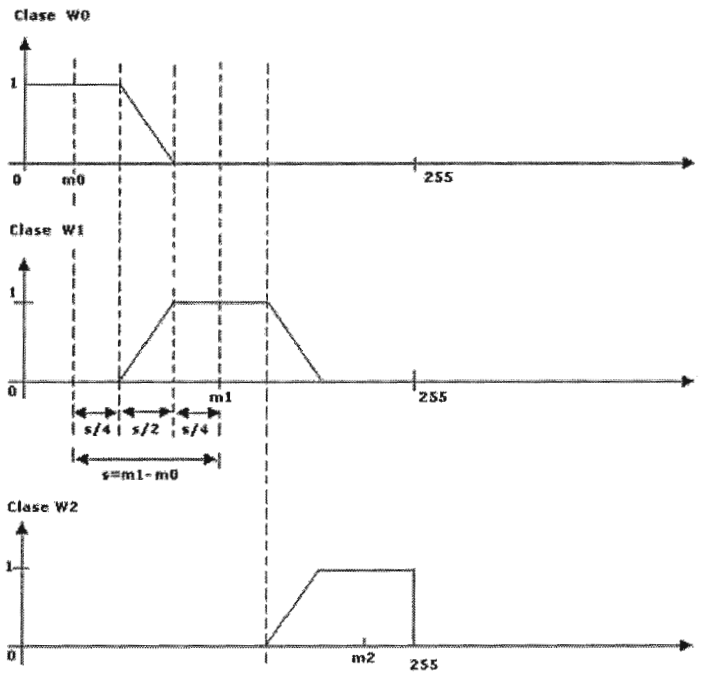



*Figura 11: Funciones de probabilidad condicional*

Segunda parte: preclasificación

1. Se aplica la función discriminante de Bayes: $p(\omega_k|x) \approx p(\omega_k)p(x|\omega_k)$
2. Como no se conoce la probabilidad de ocurrencia de clases $p(\omega_k)$, inicialmente se suponen como equiprobables.
3. Se analiza cada uno de los píxeles de la imagen $x$, clasificando a cada píxel de acuerdo a los valores máximos de probabilidad a posteriori $\arg\max[p(\omega|x)]$.

**Clasificación Bayesiana**

En esta etapa, se aplica nuevamente el proceso bayesiano anterior. En este caso, se actualizan las probabilidades a priori $p(\omega_k)$. Esta actualización se lleva a cabo dividiendo el número de píxeles que pertenecen a cada clase entre el total de píxeles de la imagen, empleando los resultados de la preclasificación.

Después de realizar la actualización de probabilidades, se clasifica cada uno de los píxeles de la imagen utilizando la misma función bayesiana de probabilidad condicional. Durante el proceso de pre-clasificación y clasificación, el nivel de gris de cada píxel se proyecta sobre cada una de las funciones para determinar la probabilidad condicional de que el píxel pertenezca a la clase $w_k$. El píxel analizado pertenecerá a la clase con máxima probabilidad a posteriori.

### 6.2.12 Support Vector Machina (SVM).

El SVM, es un método de clasificación binaria, esto quiere decir que en principio, sólo será posible clasificar cada una de las muestras como perteneciente a una de las dos clases existentes, que se pueden etiquetar como: {−1,1}.

Este sistema, consta de una fase de entrenamiento de la máquina SVM y de una fase de reconocimiento que se basa en los datos calculados en entrenamiento. Para dicho entrenamiento, el sistema se basa en un conjunto de muestras de entrenamiento $x_i$, a los que corresponde una clasificación $v_i$ (1 ó −1).

Una vez se tiene el conjunto de entrenamiento, el método se basa en maximizar el margen o la distancia entre las muestras más cercanas de ambas clases, para que la separación entre clases sea máxima.

Así pues, el objetivo perseguido por el método SVM lineal en el caso separable, consistirá en encontrar aquel hiperplano que maximice el margen entre las dos clases. Esta condición puede expresarse con las inecuaciones siguientes:

$x_i w + b \geq 1 \; para \; y_i = +1 \;\; (1)$

$x_i w + b \leq -1 \; para \; y_i = -1 \;\; (2)$

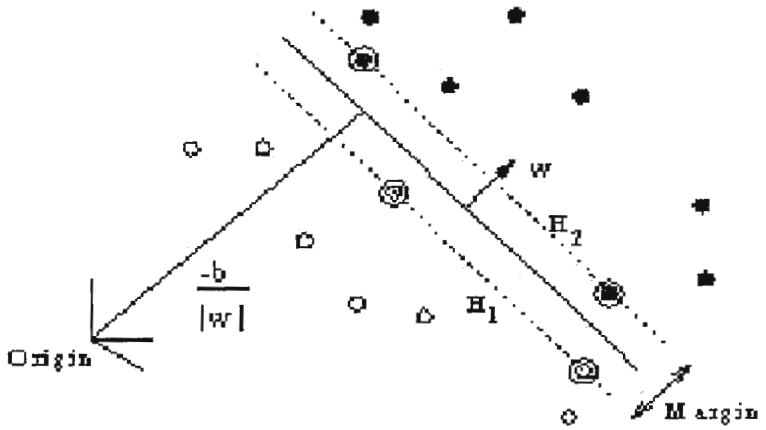



*Figura 12: SVM lineal.*

Donde *w* es el vector normal a la superficie de separación interclases.

Ahora, considérense los puntos para los que en las inecuaciones (1) y (2), se cumple la condición de igualdad (afirmar que dichos puntos existen, es equivalente a escalar los valores de b y $\|w\|$ , de forma que la distancia punto-hiperplano sea la exigida). Como vemos en la figura, la distancia entre

hiperplanos (margen a maximizar) $abs(H_1 - H_2)$ será $\frac{2}{\|w\|}$. Con lo que el objetivo será minimizar $\|w\|^2$.

Utilizando la formulación Langragiana, la relación entre las muestras aparecerá sólo en forma de productos escalares.

$$Lp \equiv \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i y_i (x_i w + b) + \sum_{i=1}^{l} \alpha_i$$

El problema se reduce a minimizar Lp, sujeto a que $\alpha_i \geq 0$ para todo $i$, y además las derivadas de Lp, respecto a cada uno de los $\alpha_i$ se anulan.

Lo anterior representa un problema convexo cuadrático, cuya resolución mediante métodos numéricos, permitirá obtener los valores de los multiplicadores de lagrange.

Una vez entrenada la máquina SVM, se pasa a la fase de decisión, es decir a la clasificación de una nueva muestra como perteneciente a una de las dos clases posibles {1.-1}. Puesto que los puntos del hiperplano central H cumplen que: $xw + b = 0$, se clasifican las muestras, según estén a un lado o a otro del hiperplano, o lo que es lo mismo, según el signo de la expresión: $\text{sgn}(xw + b)$.

Para el caso de sistemas no separables, se añade una penalización por los errores cometidos a la función objetivo.

### 6.2.13 Hidden Markov Models (HMM).

Los modelos ocultos de Markov (HMM) son un conjunto de modelos estadísticos utilizados para caracterizar las propiedades de una señal. Los HMMs consisten en 2 procesos interrelacionados:

1. Una proceso no observable, donde existen un conjunto finito de estados, una matriz probabilística de transición de estados y una distribución inicial de estados.
2. Un conjunto de funciones densidad basadas en la probabilidad de cada estado.

Los modelos ocultos de Markov han sido utilizados exitosamente en el reconocimiento del habla y de la escritura; con estos dos antecedentes, se esta impulsando actualmente el uso de los HMMs al reconocimiento facial.

Los elementos de un HMM son los siguientes:

- Un conjunto de cardinalidad $N$ de estados en el modelo; de tal forma que si $S$ es el conjunto de estados, este se puede representar como: $S=\{S_1,S_2,...,S_N\}$. El estado del modelo en un instante $t$ esta dado por: $q_t \in S, 1 \le t \le T$, donde $T$ es la longitud de una secuencia de observaciones (número de frames).
- $M$, es el número de diferentes símbolos observables; si $V$ es el conjunto de todos los símbolos posibles observables (también llamado *codebook* del modelo), entonces: $V=\{v_1,v_2,...,v_M\}$
- $A$, es la matriz de probabilidad de transición de cada estado; por ejemplo: $A=\{a_{ij}\}$ donde: $a_{ij}=P[q_{t-1}=S_i]1 \le i,j \le N$, con la condición: $0 \le a_{ij} \le 1$ y $\sum_{j=1}^{N} a_{ij}=1, 1 \le i \le N$
- $B$, la matriz de probabilidad de observación de un símbolo, por ejemplo: $B=\{b_j(k)\}$ donde $b_j(k)=P[O_t=v_k \mid q_t=S_j]$, $1 \le j \le N, 1 \le k \le M$,

donde $O_t$ es la probabilidad de observación de un símbolo en el instante $t$.

- $\Pi$, la distribución inicial de estados; por ejemplo: $\Pi = \{\pi_i\}$ donde: $\pi_i = P[q_1 = S_i], 1 \le i \le N$
- En notación corta, un modelo oculto de Harkov puede definirse como la tripla: $\lambda = (A, B, \Pi)$

En la teoría de rostros; una imagen frontal esta representada por regiones significantes (pelo, frente, ojos, nariz, boca) las cuales aparecen en un orden natural de arriba hacia abajo; cada una de estas regiones es asignada a un estado de izquierda a derecha de forma continua en un HMM tal como lo muestra la siguiente figura:

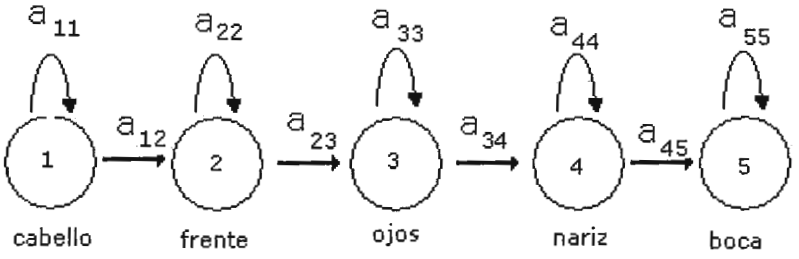



*Figura 13: Estados de un rostro según Modelos Ocultos de Harkov*

## 6.2.14 Boosting & Esenmble.

La idea de la técnica Boosting es combinar una serie de reglas simples para formar un mecanismo de deducción más complejo.

Considere los elementos: $h_1, h_2, ..., h_T$ una serie de hipótesis; y considere la función: $f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$ el ensamble de las hipótesis.

En la ecuación anterior, $\alpha_t$ denota el coeficiente con el cual el miembro $h_t$ es combinado; ambos valores $\alpha_t$ y $h_t$ son aprendidos en la técnica de "Boosting".

Las estrategias de Boosting ha sido utilizada exitosamente en varias aplicaciones del mundo real, por ejemplo para redes neuronales diseñadas para OCR; como clasificadores de tumores, entre otros; para una lista de aplicaciones desarrolladas se puede consultar: http://www.boosting.org

La idea detrás de esta técnica es emplear de manera secuencial un algoritmo con poco conocimiento basado en una serie de pesos determinados por un conjunto inicial de imágenes de entrenamiento; aunque los clasificadores individuales efectúan un reconocimiento muy leve de características, el ensemble puede proveer un clasificador más sólido. Viola y Jones construyeron el primer detector de rostros utilizando el AdaBoost lo cual fue considerado un giro dramático en la investigación de rostros.

El algoritmo AdaBoost es presentado a continuación:

1. Entrada: $S = \{(x_1, y_1),...,(x_N, y_N)\}$, con un número de iteraciones $T$
2. Inicializar: $d_n^{(1)} = 1/N$ para todos $n = 1,...,N$
3. Hacer para $t = 1,...,T$
   a. Entrenar el clasificador respecto a un conjunto de entrenamiento ejemplo $\{S, d^{(t)}\}$ y se obtiene la hipótesis: $h_t = \lambda(S, d^{(t)})$
   b. Calcular el error del entrenamiento de pesos $\varepsilon_t$ de $h_t$:

$$\varepsilon_t = \sum_{n=1}^{N} d_n^{(t)} I(y_n \neq h_t(x_n))$$

c. Fijar: $\alpha_t = \frac{1}{2}\log\frac{1-\varepsilon_t}{\varepsilon_t}$

d. Actualizar los pesos: $d_n^{(t+1)} = d_n^{(t)} \exp\{-\alpha_t y_n h_t(x_n)\}/Z_t$, donde $Z_t$ es una constante de normalización tal que: $\sum_{n=1}^{N} d_n^{(t+1)} = 1$

4. Salir del bloque de iteración si $\varepsilon_t = 0$ o $\varepsilon_t \geq \frac{1}{2}$ y fijar $T = t-1$

5. La salida es: $f_T(x) = \sum_{t=1}^{T} \frac{\alpha_t}{\sum_{r=1}^{T} \alpha_r} h_t(x)$

## *6.3 Enfoque: Principal Component Análisis (PCA)*

En el reconocimiento de rostros faciales; ha existido la tendencia de identificar características individuales como los ojos, la nariz, la boca, la barbilla determinando la posición y tamaño de cada una de estas características en el rostro. La dificultad de este enfoque es la dificultad de aplicarlo a múltiples vistas de un mismo rostro.

En el enfoque PCA, pretende determinar la variación de un rostro comparado contra un conjunto de imágenes ya registradas; sin embargo la comparación entre rostros no se hace basada en vectores distancia entre los principales rasgos del rostro humano como lo son la separación de ojos, el tamaño de la nariz o la boca, por mencionar unos ejemplos.

El análisis de Eigenrostros o también llamado de Componentes Principales (PCA) es una herramienta la cual proviene del "análisis multivariante"; esta técnica se remonta hacia el año 1901. La proyección de datos sobre el subespacio de Componentes Principales es conocida también como la

transformada Hotelling o transformada de Karhunen-Loève; lo que se busca con esta técnica es aplicar una transformada lineal a los datos en un espacio de dimensión N; la transformación se define como aquella que diagonaliza la matriz de covarianza, lo cual genera un nuevo sistema de coordenadas en un espacio de dimensión inferior M, donde M << N y los datos se encuentran descorrelacionados, es decir que las varianzas se concentran a lo largo de los ejes de referencia.

En el método PCA una imagen $I(x,y)$ se define como un arreglo bidimensional de nxn = N píxeles, donde cada píxel tiene el valor de 0 a 255[9]; por lo tanto un rostro humano puede ser modelado como una matriz bidimensional y luego descomponerse en un vector; por ejemplo considere una imagen de 128x128 píxeles, esta puede descomponerse en un vector de longitud 16,384.

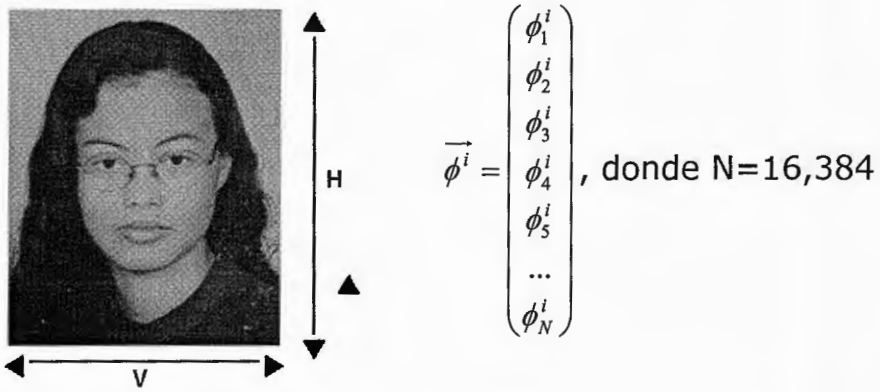



*Figura 14: Representación vectorial de una imagen*

Este vector del rostro original pertenece a un espacio, llamado espacio de imagen, y es donde se encuentran todas las imágenes cuya dimensión es

[9] El problema se aborda inicialmente con imágenes de rostros en escala de grises; sin embargo este mismo enfoque puede ser aplicado en imágenes a color.

HxV píxeles. Este espacio no es el óptimo para describir un rostro, por lo que se pretende construir un espacio que describa mejor estos rostros, los vectores básicos de este espacio son llamados "Principales Componentes".

Los eigenvectores representan la variación entre los rostros humanos identificables en el sistema; donde cada imagen registrada contribuye más o menos en cada eigenvector. La imagen resultante del cálculo de los eigenvectores es conocida como eigenrostro.

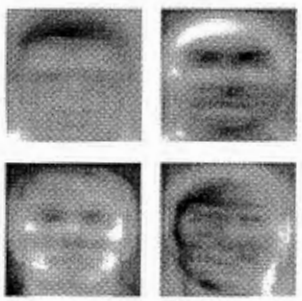


*Figura 15: Ejemplos de Eigenrostros*

Cada rostro individual se puede representar en términos de una combinación linear de eigenrostros; de hecho una multitud de imágenes pueden ser reconstruidas a partir de una suma ponderada de una colección pequeña de eigenrostros.

Este espacio óptimo para representar rostros es menor en cuanto a dimensiones que el espacio de las imágenes; si M es el número de eigenrostros representativos y $N^2$ entonces $M << N^2$.

La comparación por tanto se realiza tratando de medir la diferencia o cambios de la de la imagen desconocida sobrepuesta sobre una imagen que

representa la contribución que cada imagen de personas reconocidas hace sobre una imagen promedio.

En lenguaje matemático se buscan los componentes principales de la distribución de las imágenes o los eigenvectores de una matriz de covarianza conteniendo un conjunto de imágenes las cuales son tratadas como un vector en un espacio dimensional muy grande. Los eigenvectores son ordenados y para cada uno se cuantifica la cantidad de variación que ocurre con la imagen de un rostro en particular.

Los pasos necesarios en el proceso del reconocimiento facial mediante el enfoque PCA son los siguientes:

1. Entrenamiento.
    a. Adquirir un conjunto de rostros humanos los cuales representan las personas que el prototipo biométrico utilizará para entrenarse y reconocerlos.
    b. Calcular los Eigenrostros del conjunto de rostros humanos, y seleccionar únicamente aquellas M imágenes que poseen los más altos valores Eigenvalores. Por ejemplo, si el prototipo biométrico debe identificar 100 rostros humanos, se calcularan los Eigenrostros de cada uno pero para efectos de eficiencia en el algoritmo se pueden utilizar únicamente los 51 Eigenrostros con mayor valor de Eigenvalores (se eligen los Eigenrostros más representativos obtenidos del conjunto completo de rostros que serán identificables).
    c. Calcular la distribución M-dimensional para cada imagen que será identificada por el prototipo biométrico, esto se

hace proyectando cada imagen en las M más representativas Eigenrostros.

2. Identificación.
    a. Calcular el conjunto de pesos de la imagen desconocida en las M Eigenrostros seleccionadas en el paso del entrenamiento. Se debe proyectar la imagen desconocida en cada una de las M Eigenrostros.
    b. Determinar si la imagen desconocida forma parte del conjunto de imágenes del prototipo biométrico cuantificando las variaciones obtenidas.
    c. Si el rostro es desconocido en varios procesos de identificación, cuantificar su variación contra los M Eigenrostros representativos y reajustar pesos y/o patrones.

*Figura 16: Ejemplos de Eigenrostros representativos*

### *6.4 PCA vs ICA*

Los primeros sistemas de reconocimiento basados en apariencia utilizaron el análisis de componentes principales (PCA) como técnica de extracción de características para reducir la dimensión de los modelos o clases de objetos; sin embargo, recientemente algunos investigadores tienen preferencias sobre el método de extracción de características de componentes independientes (ICA).

El método PCA y el método ICA utilizan subespacios reducidos para representar el conjunto de rostros, de allí que están muy relacionados con la teoría de Eigenvectores y Eigenvalores.

La principal ventaja del enfoque PCA es utilizar un subespacio más representativo y equivalente donde se pueden identificar cada uno del conjunto de rostros originales que forman parte de un espacio mucho mayor.

La mayor complicación de los sistemas biométricos de Identificación de rostros viene derivada por la elevada dimensión del espacio de características de la imagen; por ejemplo, en una imagen de 128x128 píxeles la dimensión del espacio sería de 16.384 componentes. Debido a esto, es necesario el uso de técnicas que permitan reducir la dimensión del espacio en el que se realiza el análisis. La técnica más utilizada para llevar a cabo esta reducción es la del análisis en componentes principales (PCA).

El análisis de componentes independientes ICA parte de un vector aleatorio multidimensional $N^2$ (el espacio de las imágenes) y efectúa una transformación lineal que minimiza la dependencia estadística entre sus

funcionamiento sobretodo debido a problemas inherentes relacionados con los tamaños de las bases de datos; no obstante, el interés de las agencias estatales e incluso del sector financiero es alto, estimulando el alto nivel de los esfuerzos de desarrollo.

Dado este paso del desarrollo, es probable que en el futuro los computadores personales con multimedia reconozcan a sus usuarios vía una cámara fotográfica.

### 6.5.1 Introducción.

El reconocimiento mediante imágenes faciales 2D es una de las alternativas biométricas más baratas y cuenta con la ventaja de ser no intrusiva. Sin embargo, sus tasas de reconocimiento se ven limitados porque las imágenes de caras no son patrones tan estables como el ADN o las huellas dactilares.

Los productos de este tipo trabajan con varias imágenes de cada usuario y el proceso de reconocimiento está regido por una serie de reglas que conducen a una identificación efectiva. Tanto es así que algunos usan tecnología de redes neuronales propias de un esquema de inteligencia artificial que básicamente adquieren conocimiento de la experiencia.

De esta manera, siguiendo un proceso de aprendizaje, un sistema biométrico de estas características puede ir reduciendo sistemáticamente el rango de análisis de variantes faciales que recorre en la base de datos para encontrar similitudes con la cara escaneada.

### 6.5.2 Requerimientos de Hardware y Software.

El prototipo del Sistema Biométrico para la identificación de Rostros Humanos, funcionara en Plataformas Abiertas y Cerradas, esto debido a que se tiene como un objetivo principal la portabilidad del mismo. Los esquemas de licenciamiento quedaran sujetos a disponibilidad por parte del usuario final, liberando de esta manera los costos de implementación en concepto de Sistema Operativo base.

Por otro lado, para efectos de prueba y demostración de portabilidad, se plantea el uso de un sistema operativo específico para cada modalidad de licenciamiento (abierta/cerrada). Como base para los sistemas operativos Cerrados, se utilizara Windows™ en sus versiones 2000 o superiores, y para los Sistemas operativos Abiertos se utilizarán los sistemas Operativos FreeBSD 6.0 y Linux kernel 2.4.

El prototipo ejecutará una serie de cálculos matemáticos y análisis estadísticos de grandes números, sin mencionar el pre-procesamiento y procesamiento de imágenes digitales. Lo anterior sugiere una infraestructura robusta, ejecutándose bajo un esquema Cliente-Servidor y ubicando en el Servidor Central los algoritmos del núcleo del prototipo.

Los grandes fabricantes de Software para tecnologías biométricas, utilizan la figura de 2 Servidores en sus aplicaciones. Uno de ellos funciona como Motor de Bases de Datos y el otro como *"Servidor de Comparación"* o *"Servidor Algorítmico"*; En el primero los componentes de hardware mas exigidos son el almacenamiento en disco y el procesamiento; Mientras que en el segundo lo son: la memoria del servidor y el procesamiento.

El prototipo propuesto en el presente documento, será diseñado basándose en el esquema de 2 equipos independientes; por lo tanto se requerirá como hardware 2 PC con no menos de 20 GB libres en Disco Duro y un procesador de 2.0 Ghz o superiores, y en memoria RAM 512 MB.

### 6.5.3 Motor de Bases de Datos.

Como plataforma de Bases de Datos se utilizará MySQL, algunas características de este producto se muestran a continuación:

| | |
|---|---|
| **Motor de Base de Datos:** | MySQL |
| **Empresa:** | MySQL AB |
| **Ultima Versión:** | 5.0.25 |
| **Sistema Operativo:** | Multiplataforma |
| **Genero:** | RDBMS |
| **Licencia:** | GPL o Uso comercial |
| **Sitio web:** | www.mysql.com |

MySQL es uno de los Sistemas Gestores de bases de Datos (SQL) más populares desarrolladas bajo la filosofía de código abierto. La desarrolla y mantiene la empresa MySQL AB pero puede utilizarse gratuitamente y su código fuente está disponible.

Inicialmente, MySQL carecía de elementos considerados esenciales en las bases de datos relacionales, tales como integridad referencial y transacciones. A pesar de ello, atrajo a los desarrolladores de páginas web con contenido dinámico, justamente por su simplicidad; aquellos elementos faltantes fueron llenados por la vía de las aplicaciones que la utilizan.

Poco a poco los elementos faltantes en MySQL están siendo incorporados tanto por desarrollos internos, como por desarrolladores de software libre.

Entre las características disponibles en las últimas versiones se puede destacar:

- Amplio subconjunto del lenguaje SQL. Algunas extensiones son incluidas igualmente.
- Disponibilidad en gran cantidad de plataformas y sistemas.
- Diferentes opciones de almacenamiento según si se desea velocidad en las operaciones o el mayor número de operaciones disponibles.
- Transacciones y claves foráneas.
- Conectividad segura.
- Replicación.
- Búsqueda e indexación de campos de texto.

Licencia a Utilizar:

La licencia GPL de MySQL, la cual obliga a distribuir cualquier producto derivado (aplicación) bajo esa misma licencia. Por otro lado si se desea incorporar MySQL en el producto final pero no se desea distribuirlo bajo licencia GPL, se puede adquirir la licencia comercial de MySQL que permite hacer justamente eso, aunque para el desarrollo del prototipo, este no es el caso.

### 6.5.4 Lenguaje de Programación.

Python es un lenguaje de programación interpretado e interactivo, capaz de ejecutarse en una gran cantidad de plataformas. Fue creado por Guido van Rossum en 1990. Python es habitualmente comparado a TCL, Perl, Scheme, Java y Ruby. Actualmente, Python se desarrolla como un proyecto

de código abierto, administrado por la Python Software Foundation. La última versión estable del lenguaje es actualmente (Marzo de 2006) la 2.4.3. Guido van Rossum, más conocido como Guido, creó Python, un lenguaje de programación de scripting, la "oposición leal" a Perl, lenguaje con el cual mantiene una rivalidad amistosa. Los usuarios de Python consideran a éste mucho más limpio y elegante para programar. Python es un lenguaje interpretado, lo que ahorra un tiempo considerable en el desarrollo del programa, pues no es necesario compilar ni enlazar. El intérprete se puede utilizar de modo interactivo, lo que facilita experimentar con características del lenguaje, escribir programas desechables o probar funciones durante el desarrollo del programa. También es una calculadora muy útil.

Uno de los alcances de este proyecto es desarrollar un prototipo portable desarrollado con herramientas que no posean limitantes en licenciamiento que puedan encarecer o dificultar el desarrollo del mismo prototipo o de proyectos futuros basados en la presente investigación; para lograr tal propósito se ha evaluado utilizar las siguientes herramientas según sea el caso:

- Python: Lenguaje principal de programación en el proyecto, utilizado ampliamente en la comunidad OpenSource. [10]
- Python PIL: Extensiones de Python para el tratamiento de imágenes. [11]
- Python SciPy: Extensiones de Python para aplicaciones científicas. [12]

---

[10] http://www.python.org, http://code.enthought.com/enthon/

[11] http://www.pythonware.com/products/pil/

[12] http://www.scipy.org

- Python Numeric: Extensiones de Python para problemas de Algebra Lineal y vectores. [13]
- Octave: Interprete de instrucciones matemáticas, muy similar al excelente producto comercial Matlab™.

La ventaja principal de la utilización de los programas y librerías anteriormente mencionadas radica en su capacidad para la lectura de imágenes en muchos formatos gráficos (Python PIL), la implementación de rutinas que manejan estructuras de matrices y sus operaciones; así como rutinas para el cálculo de los Eigenvalores y Eigenvectores (Python SCI y Python Numeric).

La existencia de las librerías anteriormente descritas, orienta el presente trabajo en la investigación a la representación y significado de los Eigenvectores y Eigenvalores en el campo de imágenes y la implementación del enfoque PCA de una manera comprensible y escalable a proyectos futuros.

### 6.5.5 Operación.

La operatividad del Prototipo se puede descomponer en las 2 etapas; las cuales se esquematizan en la Figura 17.

[13] http://www.numpy.org

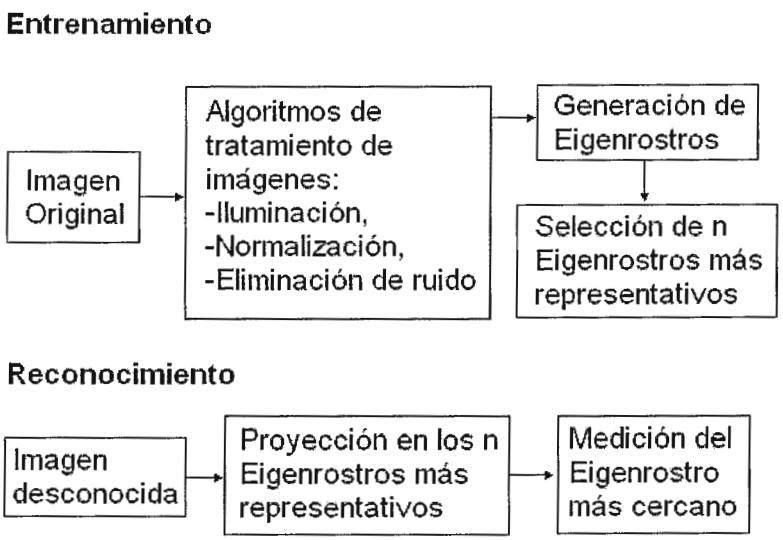



*Figura 17: Funcionamiento del Prototipo*

Se observa que existen 2 momentos cruciales de funcionamiento:

1. Entrenamiento del sistema, donde a partir de un conjunto de fotografías (en el caso del prototipo se trabajaran con un máximo de 20 fotografías) se aplican algoritmos de tratamiento de imágenes y se generan los *n* eigenrostros mas representativos (el número de eigenrostros será un parámetro modificable en el prototipo); adicionalmente a este proceso, se almacenará en una base de datos información asociada con la fotografía la cual será la siguiente:
   - Nombre de la persona
   - Código de identificación

2. Reconocimiento de una persona, donde una nueva fotografía no considerada en el conjunto de entrenamiento del paso 1, será proyectada a los *n* eigenrostros mas representativos para determinar si es un usuario válido o no; posteriormente en caso de que sea un usuario válido, será buscada en la base de datos para traer los datos asociados con esa persona.

### 6.5.6 Algoritmos de comparación

Los algoritmos que se utilizarán e implementarán en el presente prototipo son los siguientes:

- Algoritmos de lectura de imágenes en diferentes formatos: el prototipo debe ser capaz de procesar los siguientes tipos de imágenes: JPG, GIF, PNG y TIFF.

- Algoritmo de conversión a escala de grises: Se empleará si alguna de las fotografías se encuentra a colores.

- Algoritmo de normalización de escala de grises: Se empleará para normalizar la intensidad de la imagen a nivel de grises.

- Algoritmo de filtro Gaussiano: Se empleará antes de trabajar con la imagen; este filtro es recomendado para disminuir el ruido y enriquecer la información presente en una imagen que va a ser sometida a procesos de extracción de información.

- Algoritmo de cálculo de Eigenvectores y Eigenvalores: Algoritmos para cálculo de los eigenvectores y eigenvalores de una matriz cualquiera.

- Algoritmo para cálculo de Eigenrostros: Algoritmo adaptado a un conjunto de rostros para encontrar aquellos más representativos; este algoritmo involucra el trabajo con la matriz de covarianza de rostros, lo cual implicará un problema de complejidad computacional.

- Algoritmo de comparación y determinación de la cercanía de un rostro con los *n* Eigenrostros más representativos: Este algoritmo se empleará en la etapa de reconocimiento donde es necesario proyectar la imagen desconocida y determinar si puede o no ser reconstruida a partir de los eigenrostros; existen varios métodos para determinar la aproximación de una proyección a los eigenrostros, entre ellas la distancia euclidiana la cual se empleará en el presente prototipo.

# 7. MARCO TEORICO INICIAL

## *7.1 Matriz de Covarianza*

El análisis de la covarianza es una técnica estadística, la cual utiliza un modelo de regresión lineal múltiple.

El objetivo de la matriz de covarianza es comparar los resultados obtenidos en diferentes grupos de una variable cuantitativa pero corrigiendo las posibles diferencias existentes entre los grupos en otras variables que pudieran afectar el resultado (covariantes).

En el estudio conjunto de dos variables, lo que interesa principalmente es saber si existe algún tipo de relación entre ellas. Esto se ve gráficamente con el diagrama de dispersión.

La covarianza $S_{xy}$ de dos variables aleatorias $x$ y $y$ se define como:

$$S_{xy} = \sum_{i-1}^{n} \sum_{j-1}^{k} \frac{(x_i - \bar{x})(y_i - \bar{y})(n_{ij})}{n}$$

- Si $S_{xy} > 0$ hay dependencia directa (positiva), es decir, a grandes valores de $x$ corresponden grandes valores de $y$.
- Si $S_{xy} = 0$ las variables están no correlacionadas, es decir no hay relación lineal.
- Si $S_{xy} < 0$ hay dependencia inversa o negativa, es decir, a grandes valores de $x$ corresponden pequeños valores de $y$.

Las propiedades de esta medida estadística son las siguientes:

- Si todos los valores de la variable $x$, le sumamos una constante $k$, y a todos los valores de la variable $y$ le sumamos una constante $k$, la covarianza no varía.
- Si a todos los valores de una variable $x$ los multiplicamos por una constante $k$ y a todos los valores de la variable $y$ los multiplicamos por una constante $k$, su covarianza queda multiplicada por el producto de las constantes.
- Si se tienen dos variables $x, y$ con la covarianza $S_{xy}$, y las transformaciones lineales de las variables de la forma: $z = ax + b, t = cy + d$ la nueva covarianza se relaciona con la anterior de la forma: $S_{zt} = acS_{xy}$.

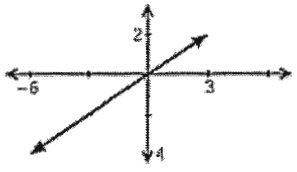
## *7.2 Independencia Lineal*

Un conjunto de vectores $\forall x, x_i \in C^n : \{x_1, x_2, ..., x_n\}$, es linealmente independiente si: $c_1 x_1 + c_2 x_2 + ... + c_n x_n = 0$ solo cuando $c_1 = c_2 = ... = c_n = 0$; por ejemplo, considere los siguientes vectores:

$x_1 = \begin{pmatrix} 3 \\ 2 \end{pmatrix}$, $x_2 = \begin{pmatrix} -6 \\ -4 \end{pmatrix}$, dichos vectores no son linealmente independientes ya que un vector se puede expresar como una constante multiplicada por otro:

$2x_1 + x_2 = 0$ es decir, no se cumple la condición $c_1 = c_2 = ... = c_n = 0$, ya que $c_1 = 2, c_2 = 1$.

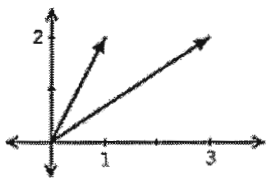
Gráficamente se puede concluir el porqué no son linealmente independientes:



Si al contrario, se consideran los siguientes vectores:

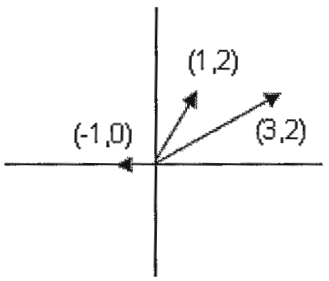
$x_1 = \begin{pmatrix} 3 \\ 2 \end{pmatrix}$, $x_2 = \begin{pmatrix} -1 \\ -2 \end{pmatrix}$, estos son efectivamente linealmente independientes ya que: $c_1 x_1 + c_2 x_2 = 0$ únicamente cuando $c_1 = 0 \wedge c_2 = 0$.

A continuación se presenta la gráfica de los dos vectores anteriores los cuales son linealmente independientes:




El problema de determinar si 2 o más vectores son linealmente independientes no es un problema sencillo, incluso si dichos vectores están en un espacio bidimensional; considere el siguiente ejemplo:

$x_1 = \begin{pmatrix} 3 \\ 2 \end{pmatrix}$, $x_2 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$, $x_3 = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$, cuya gráfica es la siguiente:




Dichos vectores, aparentan no guardar ninguna relación entre sí, sin embargo se pueden expresar con la siguiente función, la cual confirma que no son linealmente independientes.

$$x_1 - x_2 + 2x_3 = 0$$

## *7.3 Vectores Espacios*

Un vector espacio, es un conjunto *V* el cual cumple las siguientes condiciones:

- $x+y=y+x$ para cada $x,y$ en *V*.
- $x+y+z=x+y+z$ para cada $x,y,z$ en *V*.
- Existe un único vector cero, de tal forma que $x+0=x$ para cada $x$ en *V*.
- Para cada $x$ en *V*, existe un único vector $-x$ tal que $x+(-x)=0$
- $1x=x$
- $(c_1c_2)x=c_1(c_2x)$ para cada $x$ en *V*, $c_1, c_2$ en *C*.
- $c(x+y)=cx+cy$ para cada $x,y$ en *V*, $c$ en *C*.
- $(c_1+c_2)x=c_1x+c_2x$ para cada $x$ en *V*, $c_1, c_2$ en *C*.

Ejemplos de vectores espacio:

Considere el conjunto de números reales: *R* es un espacio cerrado para las operaciones de la adición y multiplicación; cada número tiene su inverso aditivo, y existen las propiedades conmutativas, asociativas y distributivas.

Considere el conjunto de números complejos: *C* este conjunto también goza de las mismas propiedades que los números reales.

El conjunto de los números complejos se puede escribir como:

$x=(x_1,x_2,...,x_n)^T$, $y=(y_1,y_2,...,y_n)^T$;

El vector suma se define como:

$$x+y=\begin{pmatrix} x_1+y_1 \\ x_2+y_2 \\ \cdots \\ x_n+y_n \end{pmatrix}$$

El producto de un escalar por un número complejo se define como:

$$zx=\begin{pmatrix} zx_1 \\ zx_2 \\ \cdots \\ zx_n \end{pmatrix}$$

## *7.4 Subespacios*

Un subespacio es un subconjunto de un vector espacio, el cual es un vector espacio en sí mismo; este concepto se puede comprender con el ejemplo de una línea trazada desde el origen en un plano cartesiano; la línea es un subconjunto, donde al sumar 2 vectores pertenecientes a la línea, el resultado será parte de dicha línea; y al multiplicar un escalar por un vector sobre la línea el resultado será parte de la línea. Este mismo concepto puede extenderse de una línea a un plano en 2 dimensiones, 3 dimensiones y n-dimensiones.

Definición: Un subconjunto *S* de un vector espacio *V* es un subespacio de V cuando se cumplen las siguientes condiciones:

- Si $x, y$ pertenecen a *S*, también pertenece $x+y$.
- Si $x$ pertenece a *S* y $t$ es un real, entonces $tx$ pertenece a S.

Por ejemplo, el subespacio $x$ definido por $x_1+x_2+x_3+x_4=0$ constituye un subespacio de $R^4$; los siguientes vectores pertenecen a dicho subespacio:

$$\begin{pmatrix}-1\\1\\0\\0\end{pmatrix}, \begin{pmatrix}-1\\0\\1\\0\end{pmatrix} y \begin{pmatrix}-1\\0\\0\\1\end{pmatrix}$$

Un subespacio *S* se dice que tiene una dimensión, si existe una colección finita de vectores dentro del subespacio $\{s_1, s_2, ..., s_n\}$ cuando cada elemento de *S* puede escribirse como una combinación lineal de dichos vectores, es decir: para cada $\forall\, s \in S$ *existe n números reales* $\{x_1, x_2, ..., x_n\}$ *tal que* $s = x_1 s_1 + x_2 s_2 + ... + x_n s_n$

### *7.5 Bases*

Una base para $C^n$ es un conjunto de vectores que:

- Generan $C^n$
- Son linealmente independientes.

Un conjunto de n vectores linealmente independientes es una base para $C^n$; por ejemplo, considere el siguiente vector:

$$e_i = \begin{pmatrix}0\\...\\0\\1\\0\\...\\0\end{pmatrix},$$

donde 1 siempre esta en la i-ésima posición y los valores resultantes son ceros; entonces se puede definir que la base para $C^n = \{\forall i, i = [1,2,...,n] : e_i\}$

## *7.6 Teoría de Eigenvectores y Eigenvalores*

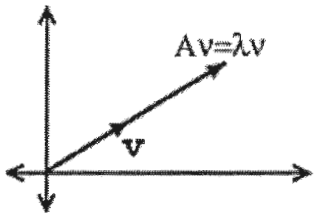
Los Eigenvectores son un conjunto de vectores asociados con un sistema linear de ecuaciones; la aplicación de los Eigenvectores y Eigenvalores es muy importante en el campo de la física y/o ingeniería, los problemas relacionados con este conjunto de vectores están relacionados con análisis de estabilidad, la física de rotación de cuerpos, momentos inerciales, ecuaciones diferenciales, compresión de imágenes, identificación por huellas dactilares, reconocimiento de huellas dactilares, reconocimiento de caracteres manuscritos, detección de características en imágenes (por ejemplo matrículas de automóviles en movimiento), detección robusta de personas u objetos móviles con fondos estáticos y en este caso con el reconocimiento de imágenes faciales.

El hablar de Eigenvectores implica la mención de los Eigenvalores los cuales no son más que el conjunto de valores correspondientes a cada vector.

Considere una matriz cuadrada M; dicha matriz tiene al menos un vector no cero el cual cumple la siguiente condición:

$$Mv = \lambda v$$

Donde, si $\lambda > 0$ significa que $Mv$ es paralelo a $v$; si $\lambda < 0$ entonces es antiparalelo; y si $\lambda = 0$ entonces $v$ se encuentra en un espacio nulo; en todos estos casos, $Mv$ es un múltiplo de $v$ , por lo tanto $M$ solo cambia la longitud de $v$ no su dirección; gráficamente se puede representar bidimensionalmente de la siguiente manera:




En la ecuación: $Mv = \lambda v$, el vector $v$ se le conoce como Eigenvector; y $\lambda$ el eigenvalor; en dicha ecuación $\lambda$ puede ser cualquier numero real o imaginario incluyendo el cero, siempre y cuando el vector $v$ no sea un vector cero.

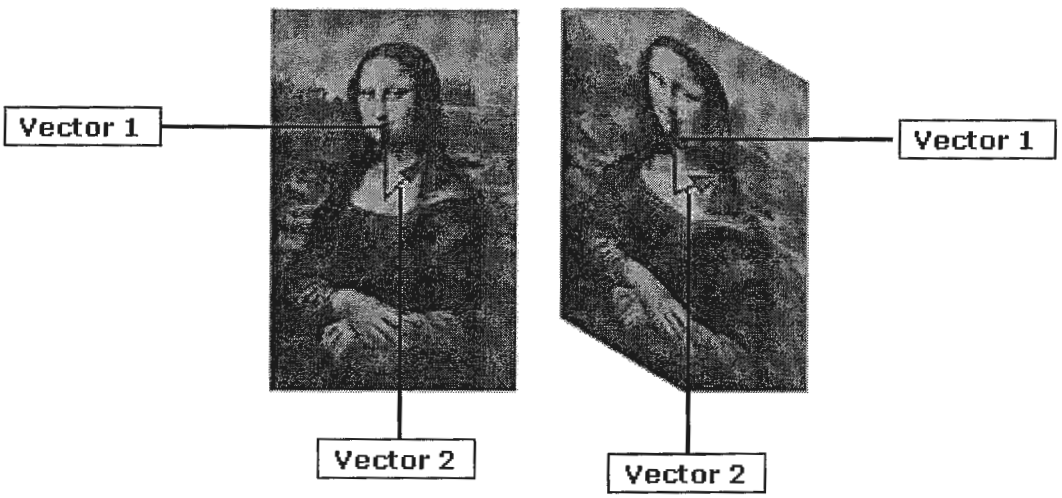



*Figura 18: Representación de un eigenvector en la trasformación de una imagen*

En la Figura 18, se trata de ejemplificar como funcionan los eigenvectores en las transformaciones de las imágenes; el cuadro de la Mona Lisa ha sido deformado de forma tal que su eje vertical no ha sido cambiado, sin embargo se han recortado las esquinas y se ha deformado el eje horizontal; bajo este escenario el vector 2 (vector que apunta desde el pecho hasta el hombre) ha cambiado de dirección y el vector 1 no ha cambiado; se puede

afirmar entonces que el vector 1 es un eigenvector de la transformación mientras que el vector 2 no lo es. Por inspección se puede afirmar que el vector 1 no cambio incluso de longitud, por lo que su eigenvalor es 1 y todos los vectores que tengan la misma dirección pero con diferente longitud serán también eigenvectores que formarán un eigenespacio de este eigenvalor.

Por ejemplo, considere la matriz:

$$A=\begin{pmatrix}3 & 0\\ 0 & -1\end{pmatrix};$$

Dicha matriz posee 2 eigenvalores: $\lambda=\{3,-1\}$, y 2 eigenvectores los cuales son:

$$v_1=\begin{pmatrix}1\\ 0\end{pmatrix},\quad v_2=\begin{pmatrix}0\\ 1\end{pmatrix}$$

De tal forma que se cumple:

$$Av=\lambda v$$

a) $$\begin{pmatrix}3 & 0\\ 0 & -1\end{pmatrix}\begin{pmatrix}1\\ 0\end{pmatrix}=3\begin{pmatrix}1\\ 0\end{pmatrix}$$

b) $$\begin{pmatrix}3 & 0\\ 0 & -1\end{pmatrix}\begin{pmatrix}0\\ 1\end{pmatrix}=2\begin{pmatrix}0\\ 1\end{pmatrix}$$

Matemáticamente existen dos clases de Eigenvectores: los Eigenvectores del lado izquierdo y los Eigenvectores del lado derecho; aunque en la mayoría de aplicaciones basta con considerar únicamente los Eigenvectores del lado derecho.

El razonamiento que conlleva las ecuaciones para encontrar los Eigenvectores y Eigenvalores por el lado derecho e izquierdo se presentan a continuación:

El efecto de descomponer una matriz cuadrada **A** en sus Eigenvalores y Eigenvectores es conocido como "Descomposición Eigen" y se basa en las siguientes ecuaciones:

Se define un eigenvector del lado derecho como un vector columna $\mathbf{X_R}$ el cual satisface la siguiente relación:

(1) $$AX_R = \lambda_R X_R$$

Utilizando la ecuación: $\lambda X_R = \lambda I X_R$, la ecuación (1) se transforma en:

(2) $$AX_R - \lambda X_R = 0$$
$$(A - \lambda_R I)X_R = 0$$

Donde $I$ es la matriz identidad:

$$I = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ 0 & 0 & \dots & \dots \\ 0 & \dots & \dots & 1 \end{pmatrix}$$, por lo tanto, $A - \lambda I$, es una matriz nueva.

Si $(A - \lambda I)X_R = 0$ para algún $v \neq 0$, entonces $A - \lambda I$ es no invertible, lo que implica que los eigenvalores del lado derecho deben tener un determinante cero.

(3) $$\det(A - \lambda_R I) = 0$$

Este determinante se transforma en una expresión polinomial (de grado $n$).

Se define un eigenvector del lado izquierdo como un vector fila $\mathbf{X_L}$ el cual satisface la siguiente condición:

(4) $$X_L A = \lambda_L X_L$$

Aplicando la operación de transposición en ambos lados de la ecuación se obtiene:

(5) $(X_L A)^T = \lambda_L X_L{}^T$

Lo cual se puede reescribir de la siguiente forma:

(6) $A^T X_L{}^T = \lambda_L X_L{}^T$

Despejando las variables y agrupando se tiene:

(7) $(A^T - \lambda_L I) X_L{}^T = 0$

Lo cual implica lo siguiente:

(8) $\det(A^T - \lambda_L I) X_L{}^T = 0$

(9) $0 = \det(A^T - \lambda_L I) = \det(A^T - \lambda_L I^T)$

(10) $= \det(A - \lambda_L I)^T$

(11) $= \det(A - \lambda_L I)$

(12) $\det(A) = \det(A^T)$

Para entender mejor el concepto y cálculo de los eigenvectores y eigenvalores, se presenta el siguiente ejemplo, considere la matriz A:

$$A = \begin{pmatrix} 3 & -1 \\ -1 & 3 \end{pmatrix},$$

$$A - \lambda I = \begin{pmatrix} 3-\lambda & -1 \\ -1 & 3-\lambda \end{pmatrix},$$

$$\det(A - \lambda I) = (3-\lambda)^2 - (-1)^2 = \lambda^2 - 6\lambda + 8$$

La resolución de la ecuación al igualarla a cero es: $\lambda = \{2,4\}$

Una vez se conocen los eigenvalores, la determinación de los eigenvectores se efectúa de la siguiente forma:

$Av = \lambda_i v$, lo cual implica: $A\begin{pmatrix} v_1 \\ ... \\ v_n \end{pmatrix} = \begin{pmatrix} \lambda_1 v_1 \\ ... \\ \lambda_n v_n \end{pmatrix}$; lo cual se transforma en un sistema de ecuaciones de $n$ incógnitas.

## *7.7 Subespacios generados por los eigenvectores*

Los eigenvectores de una matriz A, $\{v_1, v_2, ..., v_n\}$, generan el subespacio $C^n$, lo cual significa que $\{v_1, v_2, ..., v_n\}$ son linealmente independientes y se puede escribir cualquier $x \in C^n$ como: $x = \alpha_1 v_1 + \alpha_2 v_2 + ... + \alpha_n v_n$

Donde $\{\alpha_1, \alpha_2, ..., \alpha_n\} \in C$, lo cual genera las siguientes expresiones:

$$Ax = A(\alpha_1 v_1 + \alpha_2 v_2 + ... + \alpha_n v_n$$

$$Ax = \alpha_1 A v_1 + \alpha_2 A v_2 + ... + \alpha_n A v_n$$

$$Ax = \alpha_1 \lambda_1 v_1 + \alpha_2 \lambda_2 v_2 + ... + \alpha_n \lambda_n v_n = b$$

Por lo tanto: $x = \sum_i (\alpha_i v_i)$, y $b = \sum (\alpha_i \lambda_i v_i)$

# 8. PLAN DE ACCIÓN

| No. | Objetivo | Actividades | Resp. | Elemento Verificador |
|---|---|---|---|---|
| 1 | Investigación sobre Eigenvectores y Eigenvalores. | Presentar una investigación sobre los Eigenvectores, sus aplicaciones en la actualidad, las propiedades que permiten que sean utilizados en el reconocimiento de imágenes. | RM, ND | Teoría sobre Eigenvectores y PCA. |
| 2 | Diseñar e implementar algoritmos encargados de mejorar la calidad de la imagen. | Presentar de manera didáctica el conjunto de algoritmos que se emplearan posteriormente en el mejoramiento de las imágenes faciales. | RM, ND | Desarrollo de teoría de algoritmos para el tratamiento de imágenes en 2 dimensiones. |
| 3 | Diseñar algoritmo y aplicación prototipo para el cálculo de los Eigenrostros | Desarrollar la teoría de los Eigenrostros y Eigenvalores, y su aplicación a la teoría de reconocimiento facial. | RM, ND | Aplicación prototipo que calcula los Eigenrostros y los almacena en un directorio en formato JPG. |
| 4 | Diseño de algoritmo y prototipo de identificación. | Diseñar algoritmo para la identificación de un nuevo rostro basándose en la evaluación de las variaciones de los Eigenvectores | RM, ND | Cuantificación de la variación entre un rostro y las eigenrostros. |
| 5 | Prototipo de Sistema Biométrico para la identificación de Rostros Humanos | Diseño de prototipo de administración de rostros conocidos y desconocidos | RM, ND | Funcionamiento de prototipo del sistema biométrico facial. |

## Cronograma de Actividades

| ID | Actividad | Junio 06 | | | | Julio 06 | | | | Agosto 06 | | | | Septiembre 06 | | | | Octubre 06 | | | | Noviembre 06 | | | | Diciembre 06 | | | | Enero 07 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 1 | Investigación sobre Eigenvectores, Eigenvalores. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | Investigación e implementación de algoritmos para mejorar calidad de imagen | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | Algoritmos y desarrollo de módulo para Cálculo de Eigenrostros | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | Algoritmos y desarrollo de módulo para Identificación de rostros a partir de Eigenfaces | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | Primera defensa de Proyecto | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 6 | Presentación del Proyecto con observaciones incorporadas. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | Desarrollo de Prototipo de Sistema Biométrico, el cual permita identificación y administración de usuarios, manuales de usuario y técnico | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | Segunda defensa de Proyecto | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 9 | Presentación definitiva del Proyecto con observaciones incorporadas. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

# 9. COMPONENTES PRINCIPALES Y EL ANALISIS MULTIVARIANTE

## *9.1. Introducción.*

La teoría de los Componentes Principales se origina a partir del problema que se presenta en el análisis de datos multivariantes el cual consiste en reducir la dimensionalidad de un conjunto de observaciones realizadas, es decir, describir con precisión los valores de $p$ variables con un subconjunto $r$, de tal forma que $r<p$ , si se logra encontrar un conjunto de $r$ variables se dice entonces que el problema ha sido reducido en dimensiones; probablemente exista una perdida no significativa de información, sin embargo se habrá conseguido simplificar el conjunto de observaciones.

El análisis Multivariante es "la rama de la estadística que estudia las relaciones entre conjuntos de variables dependientes y los individuos para los cuales se han medido dichas variables" (Kendall). Sus métodos analizan conjuntamente $p$ variables, medidas sobre un conjunto de $n$ individuos u objetos. Una primera diferenciación entre los distintos métodos se basa en los objetivos que se persiguen, los cuales son los siguientes:

a) Simplificación estructural: se describe la información original de forma sintética o resumida; esto se logra reduciendo la complejidad del problema al condensar las $p$ variables originales en un número menor de nuevas variables creadas a partir del análisis y que contienen gran parte de la información original. Este objetivo es conocido como reducción de la dimensión y las nuevas variables creadas se denominan ejes, factores o componentes

principales. Los métodos que permiten la reducción de dimensiones son: el análisis de componentes principales (PCA), el análisis factorial (AF) y el análisis de correspondencias (AC).

b) Clasificación o agrupación: estos métodos incluyen los de agrupamiento (análisis cluster) y los de segmentación. La agrupación de individuos consiste en formar grupos de individuos homogéneos en cuanto a las *p* variables, y heterogéneos respecto a los otros grupos. La agrupación de variables busca la formación de grupos de variables similares en cuanto a comportamiento en un colectivo de objetos.

c) Análisis de interdependencia: se trata de buscar la interdependencia entre grupos de variables, sin que a priori se suponga una relación de causalidad entre ellas. El método mas conocido es el análisis de Correspondencias, que es una generalización del análisis de correspondencias Bivariante.

d) Análisis de dependencia: explica las relaciones entre grupos de variables, donde se supone que unas pueden ser causas de otras, el análisis de regresión pertenece a este grupo de métodos. En último término se trata de predecir a que grupo pertenecen nuevos objetos que no formaban parte de la información original, el análisis discriminante y la regresión logística son métodos que persiguen este objetivo.

El Análisis de Componentes Principales parte del hecho que existe *n* observaciones de *p* variables, donde dichas variables pueden ser representadas adecuadamente por un número menor de variables construidas como combinaciones lineales; dependiendo de la

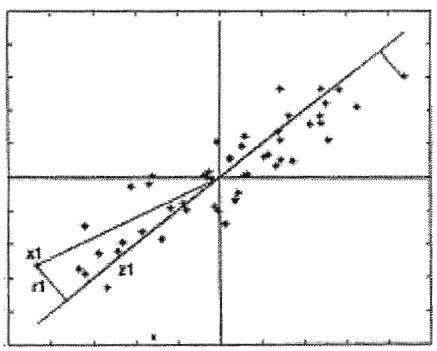
dependencia entre las variables es muy probable que menos del 20% de las nuevas variables expliquen la mayor parte de las variables originales (más del 80%).

La técnica de los Componentes Principales fue desarrollada por Hotelling (1933) sin embargo sus orígenes se encuentran en los ajustes ortogonales por mínimos cuadrados introducidos por K. Pearson (1901).

La utilidad del los Componentes Principales es la siguiente:

- Representa de forma óptima en un espacio de menor dimensión un espacio general de $p$ dimensiones. En este sentido, las componentes principales son el primer paso para identificar las posibles variables latentes o no observadas que generan datos.
- Permite la transformación de las variables originales, en general correladas, en nuevas variables incorreladas, y de esta forma facilita la interpretación de los datos.

Gráficamente el problema es fácil de presentar con una dimensión limitada a 2: considere un conjunto de puntos distribuidos sobre un plano ( $p=2$ ), y considere una línea que ha sido graficada de tal forma que pase cerca de todos los puntos:



Esta línea se dice que representa adecuadamente la información si las distancias entre los puntos originales y la proyección sobre la recta sean lo más pequeñas posibles, por ejemplo considere un punto $x_i$ y una dirección $a_1 = (a_{11},...,a_{1p})'$ definida por un vector $a_1$ cuya magnitud es la unidad; la proyección del punto $x_i$ sobre esta dirección es el escalar:

$$z_i = a_{11}x_{i1} + ... + a_{1p}x_{ip} = a_1^t x_i$$

El vector que representa la proyección es: $z_i a_1$ entre dicha proyección y el punto original se encuentra una distancia representada por $r_i$; el criterio para que la recta represente efectivamente el conjunto de puntos es minimizar dicha distancia; matemáticamente:

$$\min \sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n} |x_i - z_i a_1|^2$$

Si se proyecta cada punto sobre la recta, se forma un triángulo rectángulo donde la hipotenusa es la distancia del punto al origen, $\sqrt{(x_i' x_i)}$ y los catetos la proyección del punto sobre la recta $(z_i)$ y la distancia entre el punto y su proyección $(r_i)$; por lo que aplicando el teorema de Pitágoras se tiene:

$$x_i' x_i = (z_i)^2 + (r_i)^2$$

Sumando la expresión para todos los puntos se tiene:

$$\sum_{i=1}^{n} x_i' = \sum_{i=1}^{n} z_i^2 + \sum_{i=1}^{n} r_i^2$$

Por lo tanto, para minimizar $\sum_{i=1}^{n} r_i^2$ equivale a maximizar $\sum_{i}^{n} z_i^2$; sin embargo como las proyecciones $z_i$ son variables de media cero, la maximización de sus cuadrados equivale a maximizar su varianza, por lo tanto la metodología consistirá en maximizar la varianza de los datos proyectados.

## *9.2. Cálculos de los componentes.*

9.2.1. Cálculo de primeros componentes.

El primer componente principal es la combinación lineal de las variables originales con la varianza maximizada; estos valores se representan como un vector de la siguiente forma:

$$z_1 = Xa_1$$

La media de $z_1$ es cero como consecuencia de que las variables originales tienen media cero; la varianza esta dada por la siguiente formula:

$$\frac{1}{n} z_1' z_1 = \frac{1}{n} a_1' X' X a_1 = a_1' S a_1$$

Donde $S$ es la matriz de varianzas y covarianzas de las observaciones.

Como el objetivo es maximizar la varianza, un razonamiento lógico podría ser incrementar sin límite el módulo del vector $a_1$; lo cual nos lleva a un problema sin solución; por tal motivo es necesario imponer una restricción del vector $a_1$ de tal forma que: $a_1' a_1 = 1$; al introducir esta restricción mediante el multiplicador de Lagrange se tiene:

$$M = a_1' S a_1 - \lambda\left(a_1' a_1 - 1\right)$$

Luego la maximización viene dada derivando con respecto a los componentes $a_1$ e igualando a cero de la siguiente forma:

$$\frac{\partial M}{\partial a_1} = 2Sa_1 - 2\lambda a_1 = 0$$

Donde al despejar se tiene:

$$Sa_1 = \lambda a_1$$

Lo cual implica que $a_1$ es un vector propio de la matriz $S$ y $\lambda$ un valor propio.

Si se multiplica por $a_1'$ la ecuación anterior se tiene:

$$a_1' S a_1 = \lambda a_1' a_1 = \lambda$$

Lo cual hace concluir que $\lambda$ es la varianza de $z_1$, y como lo que se persigue al inicio es la mayor varianza, entonces se debe buscar el $\lambda$ que sea el mayor valor propio de la matriz $S$; entonces su vector $a_1$ serían los coeficientes de cada variable en el primer componente principal.

### 9.2.2 Cálculo de segundas componentes

El cálculo de las segundas componentes pretende obtener el mejor plano de proyección de las variables; este plano será definido por los vectores $a_1$ y $a_2$ donde la suma de las varianzas: $z_1 = Xa_1$ y $z_2 = Xa_2$ sea máxima.

La función objetivo es la siguiente:

$$\phi = a_1' S a_1 + a_2' S a_2 - \lambda\left(a_1' a_1 - 1\right) - \lambda\left(a_2' a_2 - 1\right)$$

La cual incorpora las restricciones de módulo unitario: $\left(a_i^{'}a_i\right)=1,\ para\ 1,2$; al derivar e igualar a cero se tiene:

$$\frac{\partial \phi}{\partial a_1} = 2Sa_1 - 2\lambda_1 a_1 = 0$$

$$\frac{\partial \phi}{\partial a_2} = 2Sa_2 - 2\lambda_2 a_2 = 0$$

Lo cual conduce al siguiente sistema:

$$Sa_1 = \lambda_1 a_1$$
$$Sa_2 = \lambda_2 a_2$$

Donde se deduce que $a_1$ y $a_2$ deben ser vectores propios de $S$ y la función objetivo realmente se reescribe como:

$$\phi = \lambda_1 + \lambda_2$$

Donde $\lambda_1$ y $\lambda_2$ deben ser los dos valores propios mayores de la matriz $S$; también $a_1$ y $a_2$ deben ser los vectores propios correspondientes a dichos valores mayores.

Otra propiedad que se verifica entre $z_1$ y $z_2$ es que la covarianza es cero porque $a_1^{'}a_2 = 0$, por lo tanto las variables $z_1$ y $z_2$ están incorreladas.

El espacio de dimensión $r$ que mejor representa a los puntos esta definido por los vectores propios asociados a los $r$ mayores valores propios de $S$. Estas direcciones son comúnmente denominadas *direcciones principales* de los datos y las variables nuevas definidas en el subespacio son denominadas *componentes principales*; en general

una matriz $X$ tiene un rango $p$ donde se pueden obtener tantos componentes principales como variables calculadas a partir de los valores propios o raíces características, $\lambda_1, \lambda_2, ..., \lambda_p$ de la matriz de varianzas y covarianzas de las variables mediante: $|S - \lambda I| = 0$ y vectores asociados: $(S - \lambda_i I)a_i = 0$.

Los términos $\lambda_i$ son reales, al ser la matriz $S$ simétrica; por este motivo además si $\lambda_j$ y $\lambda_h$ son dos raíces distintas, los vectores asociados son ortogonales.

Si se define $Z$ como la matriz cuyas columnas son los valores de los $p$ componentes en los $n$ datos, estas nuevas variables están relacionadas con las originales mediante la ecuación:

$$Z = XA$$

Donde $A'A = I$, por lo tanto el cálculo de los componentes principales equivale a la transformación ortogonal $A$ a las variables $X$ (ejes originales) para obtener nuevas variables $Z$ incorreladas entre sí; esto visualmente puede explicarse como la elección de nuevos ejes de coordenadas que coincidan con los ejes naturales de los datos.

### 9.2.3. Propiedades

Los componentes principales poseen las siguientes propiedades:

a) Conservación de la variabilidad inicial, es decir, la suma de las varianzas de los componentes es igual a la suma de las varianzas de

las variables originales y la varianza generalizada de los componentes es igual a la original.

Demostración:

$$Var(Z_h) = \lambda_h$$

La suma de los valores propios es equivalente a la traza de la matriz:

$$tr(S) = Var(x_1) + ... + Var(x_p) = \lambda_1 + ... + \lambda_2$$

$$\sum_{i=1}^{p} Var(x_i) = \sum \lambda_i = \sum_{i=1}^{p} Var(z_i)$$

La varianza generalizada se obtiene a partir del determinante de la matriz de covarianza de las variables; si se llama $S_z$ a la matriz de covarianza de los componentes, la cual es diagonal en términos de $\lambda_i$ se tiene:

$$|S_x| = \lambda_1 ... \lambda_p = \prod_{i=1}^{p} Var(z_i)$$

b) La proporción de variabilidad explicada por un componente es el cociente entre la varianza (valor propio asociado al vector propio que lo define) y la suma de los valores propios de la matriz:

Si la varianza del componente $h$ es $\lambda_h$ , la suma de las varianzas de las variables originales es $\sum_{i=1}^{p} \lambda_i$ , por lo tanto la proporción de variabilidad total explicada por el componente $h$ es $\frac{\lambda_h}{\sum \lambda_i}$ .

c) Las covarianzas entre cada componente principal y las variables originales viene expresada por el producto de las coordenadas del vector propio que define el componente por su valor propio:

$$Cov(z_i; x_1,...,x_p) = \lambda_i a_i = (\lambda_i a_{i1},...,\lambda_i a_{ip})$$

Donde $a_i$ es el vector de coeficientes de la componente $z_i$

Demostración:

Considere una matriz $p\,x\,p$ de covarianzas entre los componentes y las variables originales. Esta matriz se define como:

$$Cov(z,x) = \frac{1}{n} Z' X$$

La primera fila proporciona las covarianzas entre la primera componente y las $p$ variables originales; como $Z = XA$, al sustituir se tiene:

$$Cov(z,x) = \frac{1}{n} A' X' X = A' S = DA'$$

Donde $A$ contiene en columnas los vectores propios de $S$ y $D$ la cual es la matriz diagonal de los vectores propios; por lo tanto la covarianza entre, por ejemplo, el primer componente principal y las $p$ variables se calcula como la primera fila de $A'S$, es decir, $a_1' S$ o también $\lambda_1 a_1'$ donde $a_1'$ es el vector de coeficientes de la primera componente principal.

d) La correlación entre un componente principal y una variable $X$ es proporcional al coeficiente de esa variable en la definición del componente, y el coeficiente de proporcionalidad es el cociente entre la desviación típica del componente y la desviación típica de la variable.

$$Corr(z_i; x_j) = \frac{Cov(z_i x_j)}{\sqrt{Var(z_i)Var(x_j)}} = \frac{\lambda_i a_{ij}}{\sqrt{\lambda_i s_j^2}} = a_{ij}\frac{\sqrt{\lambda_i}}{s_j}$$

e) Las $r$ componentes principales ($r < p$) proporcionan la predicción lineal óptima con r variables del conjunto de variables $X$

f) Si se estandariza los componentes principales, dividiendo cada componente entre su desviación típica, se obtiene la estandarización multivariante de los datos originales.

Al estandarizar los componentes $Z$ por sus desviaciones típicas, se obtienen las nuevas variables:

$$Y_c = ZD^{-\frac{1}{2}} = XAD^{-\frac{1}{2}}$$

Donde $D^{-\frac{1}{2}}$ es la matriz que contienen las inversas de las desviaciones típicas de las componentes.

La estandarización multivariante de una matriz de variables $X$ de media cero se define como:

$$Y_s = XAD^{-\frac{1}{2}}A'$$

### *9.3 Análisis normado con variables correladas*

Los componentes se obtienen maximizando la varianza de la proyección, es decir:

$$M = \sum_{i=1}^{p} a_i^2 s_i^2 + 2\sum_{i=1}^{p} \sum_{j=i+1}^{p} a_i a_j s_{ij}$$

Con la restricción de $a'a = 1$; sin embargo si alguna de las variables tiene una varianza $s_i^2$ mayor que las demás este influirá de manera notable en la maximización y probablemente el primer componente principal coincida aproximadamente con esta variable. Para evitar este problema se recomienda la estandarización de las variables antes del cálculo de los componentes de tal forma que las magnitudes de los valores numéricos de las variables $X$ sean similares; adicionalmente la estandarización resuelve otro problema cuando las variabilidades de las $X$ son muy distintas ya que en este caso únicamente las variables con mayor varianza influirán en el cálculo de la primera componente.

Al realizar la estandarización de las variables, las varianzas son la unidad y las covarianzas son los coeficientes de correlación, luego la ecuación a maximizar se transforma en:

$$M' = 1 + 2\sum_{i=1}^{p} \sum_{j=i+1}^{p} a_i a_j r_{ij}$$

Donde $r_{ij}$ es el coeficiente de correlación lineal entre las variables $i$ y $j$; por lo tanto la solución depende de las correlaciones no de las varianzas.

Los *componentes principales normados* se obtienen calculando los vectores y valores propios de la matriz $R$, de coeficientes de correlación. Si se llama $\lambda_p^R$ a las raíces características de esa matriz se verifica:

$$\sum_{i=1}^{p} \lambda_i^R = traza(R) = p$$

Las propiedades de los componentes extraídos de $R$ son:

a) La proporción de variación explicada por $\lambda_p^R$ será: $\frac{\lambda_p^R}{p}$

b) Las correlaciones entre cada componente $z_j$ y las variables $X$ originales vienen dados por: $a_j^{'}\sqrt{\lambda_j}$ siendo $z_j = Xa_j$

El análisis de matriz de correlaciones o análisis normado es sugerido cuando las variables se encuentran en diferentes unidades o escalas, o cuando la variabilidad de las variables es muy marcada.

## *9.4. Interpretación de los componentes*

Si las variables tienen una alta correlación, el primer componente principal tiene todas sus coordenadas del mismo signo y puede interpretarse como un promedio ponderado de todas las variables o un factor global de "tamaño". Las componentes restantes se pueden interpretar como factores "de forma" y típicamente tienen coordenadas positivas y negativas lo cual implica una contraposición de unos grupos variables frente a otros. Estos factores de forma se escriben como

medias ponderadas de dos grupos de variables con distinto signo y contraponen las variables de un signo a las del otro.

### *9.5. Selección del número de componentes*

Existen diversas reglas que tratan de establecer la mejor manera de seleccionar el número de componentes que representan la información:

- Realizar un gráfico de $\lambda_i$ frente a $i$; donde se pretende localizar el punto donde los valores propios son aproximadamente iguales; de esta forma se persigue trabajar con aquellos componentes de un mismo tamaño y evitar los valores pequeños.
- Selección de componentes hasta cubrir un determinado porcentaje de la varianza (80% ó 90%); este método es arbitrario y puede excluir componentes que describan la "forma" de las mediciones.
- Desechar componentes inferiores a determinado limite; generalmente el límite se establece como la varianza media, $\sum \lambda_i / p$; si se esta trabajando con una matriz de correlación el valor medio de los componentes es 1 y la regla se adapta a seleccionar los valores propios mayores que la unidad.

### *9.6. Graficas de los componentes principales*

Generalmente la interpretación de un fenómeno estadístico se comprende mejor al graficar las proyecciones de las observaciones en un espacio bidimensional; la metodología es considerar 2 componentes

como dos ejes ortogonales y sobre dicho plano graficar los valores propios; por ejemplo en un plano donde los ejes ortogonales elegidos son: $a_1$ y $a_2$ las coordenadas del punto $x_i$ será: $z_{1i} = a_1' x_i$ y $z_{2i} = a_2' x_i$.

Adicionalmente a la graficación de los valores propios; la interpretación de los datos se enriquece al graficar los valores originales en el plano formado por 2 componentes ortogonales seleccionados; para graficar las variables originales se utilizan como coordenadas el coeficiente de correlación con cada uno de los ejes; el vector de correlaciones entre el primer componente y las variables originales se expresa como: $\lambda_1^{\frac{1}{2}} a_1' D$, donde $D$ es la matriz diagonal cuyos términos son las inversas de las desviaciones típicas de cada variable. La matriz de correlaciones $R_{cv}$ entre los $p$ componentes y las $p$ variables tiene como filas los términos: $\lambda_j^{\frac{1}{2}} a_j' D$ la cual se reescribe de la siguiente forma: $R_{cv} = \Lambda^{\frac{1}{2}} AD$ donde $A$ es la matriz de vectores propios, $\Lambda^{\frac{1}{2}}$ es la matriz diagonal con términos $\sqrt{\lambda_i}$; en el análisis normado como las variables se encuentran estandarizadas a varianza unidad, la correlación será: $\Lambda^{\frac{1}{2}} A$

## *9.7 Utilización de vectores propios en estructuras.*

La idea de componentes principales se extiende para la búsqueda de representaciones no lineales de los datos que expliquen las estructuras; este enfoque se utiliza cuando se sospecha que los datos pueden ordenarse en una determinada superficie en el espacio.

Los vectores propios cuyos valores propios son próximos a cero son importantes porque revelan relaciones de poca variabilidad de los datos.

Si existe una relación cualquiera no lineal entre las variables, esta relación se puede aproximar mediante una relación polinómica:

$$f(x_1,...,x_p)=\sum a_i x_i+\sum b_{ij}x_i x_j+\sum c_{ijk}x_i x_j x_k+...$$

Si se incluyen nuevas variables adicionales como $x_1^2,...,x_p^2$ o productos de variables $x_1x_2$, etc. y se extraen los componentes principales de la matriz de correlaciones entre todas estas variables, si los puntos tienen una relación no lineal ésta se identifica mediante la presencia de un valor propio próximo a cero. Este enfoque se conoce como "Componentes Principales Generalizados".

# 10. Identificación de rostros a partir de PCA

## *10.1 Diagrama general del proceso*

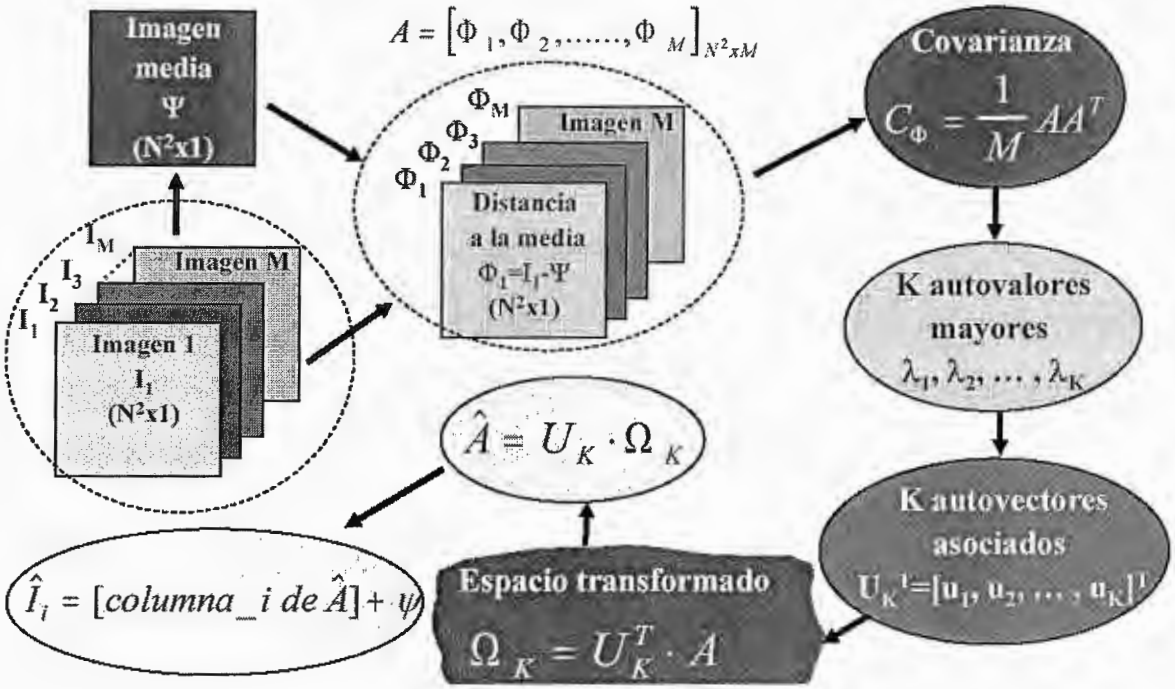
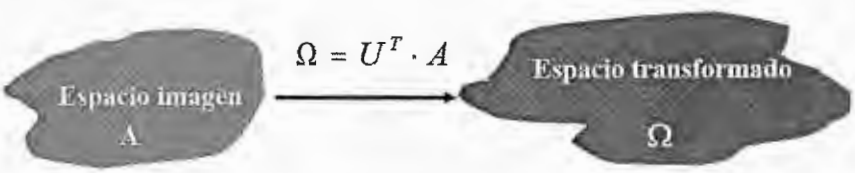



*Figura 19: Diagrama general del Proceso*




$$\Omega = [\Omega_1, \Omega_2, \ldots\ldots, \Omega_M]_{N^2 xM} = \begin{bmatrix} \Omega_{11} & \Omega_{21} & \ldots & \Omega_{M1} \\ \Omega_{12} & \Omega_{22} & \ldots & \Omega_{M1} \\ \ldots & \ldots & \ldots & \ldots \\ \Omega_{1N^2} & \Omega_{2N^2} & \ldots & \Omega_{MN^2} \end{bmatrix}$$

*Figura 20: Transformación del Espacio*

## *10.1. Introducción matemática*

La identificación de rostros a partir de los componentes principales o Eigenrostros es conocida también como la transformada de Hotelling o transformada de Karhunen-Loéve (KLT); el método se basa en la transformación lineal de los datos en un espacio de dimensión $N$, dicha transformación diagonaliza la matriz de covarianza de las muestras; al aplicar esta transformada se obtiene un nuevo sistema de coordenadas de un espacio sensiblemente menor $M$ tal que ($M << N$) donde la nueva información se encuentra descorrelacionada a lo largo de nuevos ejes de referencia; estos nuevos *Ejes Principales* son aquellos donde las muestras presentan una pequeña o mínima variación (mínima varianza).

El tratamiento de una imagen se realizará de partiendo de la siguiente definición: una imagen $I(x,y)$ se define como un arreglo bidimensional de $n\,x\,n = N$ píxeles donde cada píxel puede tener 256 valores de intensidad diferentes (8 bits); por lo tanto la información de una imagen puede ser almacenada en un vector de dimensión $N$ equivalente a $256x256 = 65{,}536$ el cual será nuestro espacio original.

La idea central del análisis de Componentes Principales (PCA) (o Karhunen-Loéve - KLT) encontrar los vectores que mejor representen la distribución de imágenes, dichos vectores presentan las siguientes características:

- Definen un "subespacio de imágenes"
- Tienen una longitud $N$ (idéntica a los vectores originales)
- Son autovectores de la matriz de covarianza; por esta razón son conocidos como Eigenrostros.
- Son ortonormales.

Dado un conjunto de muestras de imágenes de rostros en los vectores columna $\Gamma_0, \Gamma_1, ..., \Gamma_{M-1}$ se define una imagen media de las muestras al vector columna mediante la siguiente expresión:

$$\Psi = \frac{1}{M} \sum_{j=0}^{M-1} \Gamma_j \qquad (1)$$

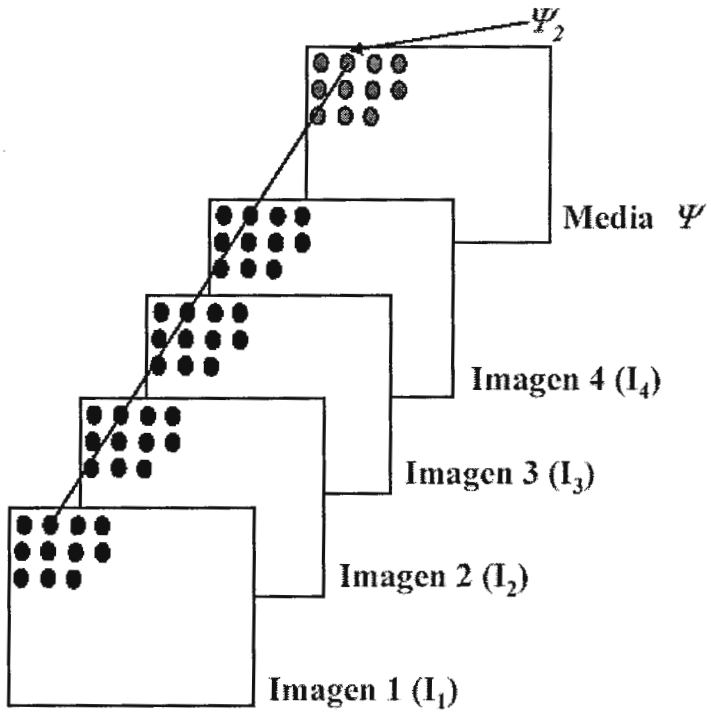



*Figura 21: Calculo de la imagen media*

Los nuevos vectores con media nula serán los siguientes:

$$\Phi_i = \Gamma_i - \Psi \qquad (2)$$

Posteriormente se procede a encontrar los vectores ortonormales $u_i$ que describirán de la mejor manera los datos; de esta forma, las imágenes originales pueden ser reconstruidas a partir de estos vectores a través de la siguiente expresión:

$$\Gamma_j = \Psi + \sum_{i=0}^{M-1} w_{ij} u_i \tag{3}$$

Donde $\{u_i\}$ es una base ortonormal del subespacio de las muestras con $i = 0, 1, ..., M-1$ es decir:

$$u_k^T . u_l = \delta_{l,k} = \begin{cases} 1 & si\ l = k \\ 0 & si\ l \neq k \end{cases} \tag{4}$$

Los coeficientes $w_{ij}$ están dados por las proyecciones sobre los eigenrostros, matemáticamente su definición es la siguiente:

$$w_{ij} = \Phi_j^T . u_i \tag{5}$$

Los vectores ortonormales $u_i$ se utilizan también para encontrar las direcciones donde las desviaciones se encuentran concentras, en otras palabras, donde los datos transformados están descorrelacionados; la siguiente ecuación $\lambda_k$ provee una medida de la cantidad de desviación en la dirección del vector $u_k$:

$$\lambda_k = \frac{1}{M} \sum_{i=0}^{M-1} \left( u_k^T . \Phi_i \right)^2 \tag{6}$$

La matriz de covarianza esta definida de la siguiente forma:

$$C = \frac{1}{M}\sum_{i=0}^{M-1}\Phi_i\Phi_i^T \tag{7}$$

Utilizando la definición de matriz de covarianza, la medida de desviación de los vectores $u_i$ se redefine de la siguiente manera:

$$\lambda_k = u_k^T C u_k \tag{8}$$

Por lo tanto los vectores $u_k$ y los escalares $\lambda_k$ son los eigenvectores y eigenvalores de la matriz de covarianzas $C$; esta matriz de covarianzas es de dimensión $N\,x\,N$ $(65{,}536\,x\,65{,}536)$ y simétrica, por lo tanto es semidefinida positiva y todos sus autovalores son no-negativos; otra característica que poseerán los autovalores es que serán pequeños y en su mayoría cero debido a la similitud que existe a nivel rostros.

La matriz de covarianza $C$ tiene un total de $N$ eigenvectores y $N$ eigenvalores, para dicho calculo es necesario procesar una matriz de $65{,}536\,x\,65{,}536$; esto es un problema muy complejo desde el punto de vista computacional por la cantidad de operaciones que debe ejecutar la computadora; adicionalmente, el número de muestras o imágenes $M$ que se procesarían es muy inferior a N ($M << N$) y generarían sólo $M$ autovectores.

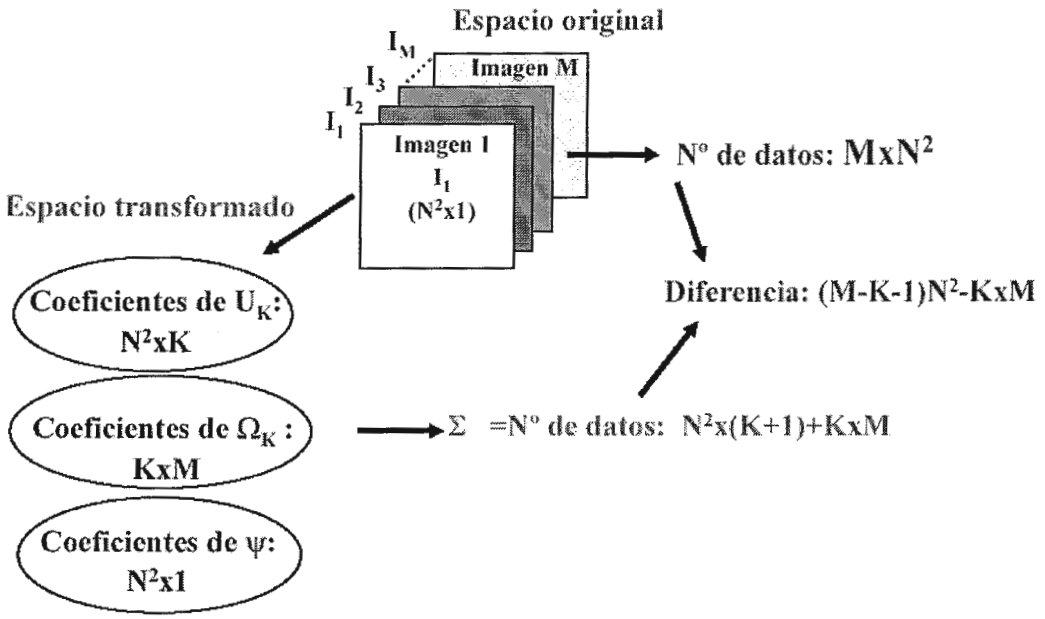



*Figura 22: Reducción de la complejidad de un problema*

Debido al problema computacional que implica el procesamiento de una matriz de dimensiones $N \, x \, N$ es necesario recurrir al método introducido por M. Turk y A. Pendtland el cual reduce el problema de $N \, x \, N$ dimensiones a uno muy inferior de $M \, x \, M$ dimensiones, este método es mucho más factible de implementar desde el punto de vista computacional (por ejemplo, si la muestra fuesen únicamente 16 rostros, se trabajaría con un problema de $16 \, x \, 16$ dimensiones en lugar de $256 \, x \, 256$.

El proceso de M. Turk y A. Pendtland supone la definición de una matriz $A$ de $N \, x \, M$ dimensiones cuyas columnas son los vectores de las imágenes con promedio nulo, es decir:

$$A = [\Phi_0 \; \Phi_1 \; \Phi_2 \; \Phi_3 \; \ldots \Phi_{M-3} \; \Phi_{M-2} \; \Phi_{M-1}] \qquad (9)$$

Por lo tanto, la matriz de covarianza se reescribe de la siguiente manera:

$$C = \frac{1}{M}\sum_{i=0}^{M-1}\Phi_i\Phi_i^T = \frac{1}{M}AA^T \tag{10}$$

La diagonalización de la matriz $A^TA$ de $M\,x\,M$ dimensiones viene expresada como la resolución de la siguiente ecuación:

$$A^TAv_i = \mu_i v_i \tag{11}$$

Con $i = 0, 1, 2, \ldots M-1$ y donde $v_i$ y los $\mu_i$ son los eigenvectores y eigenvalores de la matriz $A^TA$.

Al multiplicar la ecuación anterior por el término $\frac{1}{M\sqrt{\mu_i}}A$ se obtiene la siguiente expresión:

$$\frac{1}{M}AA^T\left(\frac{1}{\sqrt{\mu_i}}Av_i\right) = \frac{1}{M}\mu_i\left(\frac{1}{\sqrt{\mu_i}}Av_i\right) \tag{12}$$

La ecuación anterior combinada con la ecuación $C = \frac{1}{M}\sum_{i=0}^{M-1}\Phi_i\Phi_i^T = AA^T$ genera como resultado:

$$C\left(\frac{1}{\sqrt{\mu_i}}Av_i\right) = \frac{1}{M}\mu_i\left(\frac{1}{\sqrt{\mu_i}}Av_i\right) \tag{13}$$

Lo cual lleva a la conclusión que los eigenvectores y eigenvalores de la matriz $C$ se pueden obtener utilizando los eigenvectores y

eigenvalores de la matriz reducida $A^T A$ utilizando las siguientes ecuaciones:

$$u_i = \frac{1}{\sqrt{\mu_i}} A v_i \tag{14}$$

$$\lambda_i = \frac{1}{M} \mu_i \tag{15}$$

Donde el conjunto de vectores $\{u_i\}$ con $i = 0, 1, 2, \ldots M-1$ será un conjunto ortonormal de vectores siempre y cuando el conjunto de vectores $\{v_i\}$ lo sea:

Se consideran las proyecciones de los vectores $\{u_i\}$ entre sí:

$$u_l^T . u_k = \left( \frac{1}{\sqrt{\mu_l}} A v_l \right)^T . \left( \frac{1}{\sqrt{\mu_k}} A v_k \right)$$

$$u_l^T . u_k = \frac{1}{\sqrt{\mu_l \mu_k}} A^T A v_l^T . v_k \tag{16}$$

Utilizando las ecuaciones 11 y 4 en la ecuación 16 se demuestra la ortonormalidad de $\{u_i\}$

$$u_l^T . u_k = \frac{\sqrt{\mu_l}}{\sqrt{\mu_k}} v_l^T . v_k = \delta_{l,k} \tag{17}$$

## *10.2. Aproximación a través de la expansión de Karhunen-Loève truncada*

La teoría de los componentes principales define que cualquier imagen de las muestras de aprendizaje tiene una representación exacta a través de los coeficientes de proyección sobre el subespacio.

Considere una aproximación de una imagen cualquiera de las muestras de aprendizaje $\Gamma_j$ como la expansión PCA truncada:

$$\hat{\Gamma}_j = \Psi + \sum_{i=0}^{L-1} w_{ij} u_i \tag{18}$$

Con $L < M$; el error cuadrático entre la imagen original y la aproximación truncada es:

$$\left\| \Gamma_j - \hat{\Gamma}_j \right\|^2 = \left\| \sum_{i=L}^{M-1} w_{ij} u_i \right\|^2 \tag{19}$$

El error cuadrático medio se deduce de la ecuación anterior:

$$\begin{aligned} E\left[error^2\right] &= E\left[ \left\| \Gamma_j - \hat{\Gamma}_j \right\|^2 \right] \\ &= E\left[ \left\| \sum_{i=L}^{M-1} w_{ij} u_i \right\|^2 \right] \\ &= E\left[ \left( \sum_{i=L}^{M-1} w_{ij} u_i \right)^T \left( \sum_{k=L}^{M-1} w_{kj} u_k \right) \right] \\ &= E\left[ \sum_{i=L}^{M-1} \sum_{k=L}^{M-1} w_{ij} w_{kj} u_i^T . u_k \right] \end{aligned} \tag{20}$$

Utilizando la ecuación $u_l^T.u_k = \frac{\sqrt{\mu_l}}{\sqrt{\mu_k}} v_l^T.v_k = \delta_{l,k}$ (17) se llega al siguiente resultado:

$$E\left[error^2\right] = E\left[\sum_{i=L}^{M-1} w_{ij}^2\right]$$

Y aplicando la ecuación $w_{ij} = \Phi_j^T.u_i$ (5):

$$E\left[error^2\right] = E\left[\sum_{i=L}^{M-1} u_i^T \Phi_j \Phi_j^T.u_i\right]$$

$$E\left[error^2\right] = \sum_{i=L}^{M-1} u_i^T E\left[\Phi_j \Phi_j^T\right] u_i \qquad (21)$$

Teniendo en cuenta las ecuaciones 10, 13, 14 y 15 reemplazándolas en la ecuación anterior se obtiene al final:

$$E\left[error^2\right] = \sum_{i=L}^{M-1} u_i^T C.u_i$$

$$= \sum_{i=L}^{M-1} u_i^T.(\lambda_i u_i)$$

$$= \sum_{i=L}^{M-1} \lambda_i \|u_i\|^2$$

$$E\left[error^2\right] = \sum_{i=L}^{M-1} \lambda_i \qquad (21)$$

Esta última ecuación indica que el error en la aproximación es equivalente a la suma de los eigenvalores no utilizados; considerando que la transformada de Karhunen-Loève utiliza los eigenvalores más representativos se puede concluir que dicha aproximación es muy certera en términos del Error Cuadrático Medio; lo cual hace a la

transformada de Karhunen-Loève una herramienta de compresión la cual puede utilizarse para reconstruir la información con bastante exactitud únicamente con una cantidad fija de eigenvalores.

### *10.3 Clasificación de una imagen desconocida utilizando PCA.*

En la sección anterior se demostró que la aproximación de la transformada de Karhunen-Loève permite reconstruir cualquier imagen original a partir de un número finito de eigenfaces; esta aproximación se loatra a través del conjunto de coeficientes $\{w_{ij}\}$ con $i, j = 0, 1, ..., M-1$. En la teoría de estimación, el conjunto $\{w_{ij}\}$ representa el conjunto de características.

Considere el siguiente problema; se cuenta con una imagen $\Lambda$ la cual probablemente no es exactamente idéntica a ninguna de las imágenes iniciales que se utilizo en el entrenamiento y cálculo de los eigenfaces; el primer paso requerido para "identificar" dicha imagen es calcular la proyección de dicha imagen sobre el subespacio de las Eigenfaces:

$$c_i = u_i^T.(\Lambda - \Psi) \qquad (23)$$

Posteriormente se aplica el criterio del vecino más cercano para clasificar dicha imagen con alguna(s) que fue utilizada en el entrenamiento original; el criterio del vecino más cercano se aplica calculando las distancias euclideas en el espacio de las características e identificarla o asociarla con la imagen más cercana, por lo tanto es necesario calcular tantas distancias euclideas como cantidad de muestras de aprendizaje se utilizaron:

$$d(j) = \sum_{i=0}^{M-1} (c_i - w_{ij})^2 \qquad (24)$$

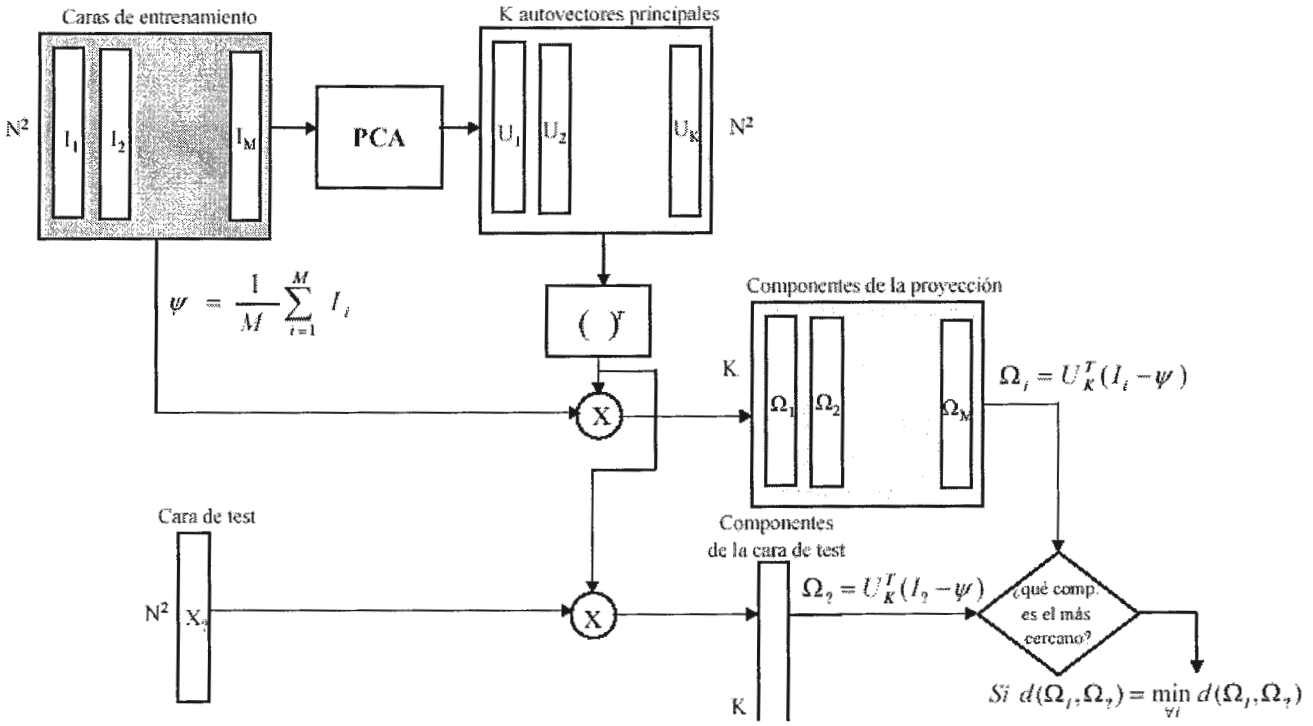



*Figura 23: Diagrama de proceso de reconocimiento*

# 11. Tratamiento de Imágenes

## *11.1. Tratamiento de imágenes y su transformación en vectores*

Para el tratamiento de imágenes se utiliza la librería: Python Imaging Library (PIL) (http://www.pythonware.com/products/pil) ; esta librería soporta la lectura de aproximadamente 30 distintos tipos de formato de archivos; los tipos de formato más importantes son los siguientes:

| Formato | Descripción |
|---|---|
| BMP | Lectura y escritura de BMP de Windows y OS/2 del tipo "1", "L", "P" o "RGB" |
| CUR | Lectura de archivos para almacenar cursores en Windows; no es soportado cursores animados. |
| DCX | Lectura de archivo contenedor de información en formato PCX definido por Intel; los tipos soportados son "1", "L", "P" o "RGB" |
| EPS | Escritura de imágenes EPS |
| FLI, FLC | Lectura de animaciones en formato Autodesk FLI y FLC. |
| FPX | Lectura de archivos Kodak FlashPix. |
| GBR | Decoder de archivos utilizados por GIMP |
| GD | Lectura de archivos no comprimidos GD. |
| GIF | Soporte para GIF87a y GIF89a |
| ICO | Lectura de archivos de iconos en Windows. |
| IM | Formato utilizado por LabEye y aplicaciones relacionadas con la librería de procesamiento de imágenes IFUNC. |
| IMT | Lectura de archivos en formato Image Tools |
| JPG | Lectura de archivos JPEG, JFIF y Adobe JPEG en formato "L", "RGB" o "CMYK". |
| MIC | Soporte para archivos en formato Microsoft Image Composer |
| MCIDAS | Lectura del formato 8-bit Mcldas |

| | |
|---|---|
| MPEG | Identificación de archivos MPEG |
| MSP | Lectura de archivos MSP de Windows. |
| PCD | Lectura de archivos PhotoCD |
| PCX | Lectura de archivos PCX conteniendo información en formato "1", "L", "P" o "RGB" |
| PDF | Escritura de archivos PDF utilizando formato JPEG o HEX. |
| PNG | Lectura de archivos PNG en formato "1", "L", "P", "RGB" o "RGBA". |
| PPM | Lectura y escritura de archivos PBM, PGM y PPM en formato "1", "L" o "RGB" |
| PSD | Archivos de Adobe Photoshop 2.5 y 3.0 |
| SGI | Lectura de archivos SGI en formato "L" y "RGB" (Este driver es experimental) |
| SUN | Archivos SUN en formato "1", "P", "L" y "RGB" |
| TGA | Lectura de archivos TGA de 24 y 32 bits sin compresión |
| TIFF | Lectura y escritura de archivos TIFF en formato "1", "L", "RGB" o "CMYK" |
| XBM | Lectura y escritura de X Bitmaps (formato "1") |
| XPM | Lectura de X Pixmaps (formato "P") con 256 o menos colores. |

La apertura de una imagen se efectúa importando el módulo en el script y utilizando la función **open** tal como se muestra a continuación:

```
import Image
foto = Image.open("192834.JPG")
```

La librería PIL procesa una imagen como un objeto el cual posee entre otras propiedades las siguientes:

- Un atributo denominado **size** la cual es una tupla que indica la cantidad de pixeles horizontales y verticales
- Una función denominada **getpixel(xy)** la cual retorna el valor del píxel en la posición xy (retorna una tupla si la imagen tiene múltiples capas).

La percepción de una imagen como una matriz bidimensional no es muy útil en la teoría de componentes principales y en general de Algebra Lineal, donde se acostumbra a trabajar con vectores; por tal motivo es necesario transformar el modelo bidimensional a un modelo unidimensional, específicamente al modelo de un vector; dicha transformación se realiza en 2 etapas:

- Transformación del objeto Image de la librería PIL en un array definido por la librería: Numerical Python (http://numpy.sourceforge.net); dicha transformación es requerida ya que esta librería implementa algoritmos eficientes para operaciones entre matrices.
- Vectorización del array.

La transformación del objeto Image se realiza utilizando el módulo pilutil de la librería SciPy; dicho módulo implementa dos funciones útiles:

- pilutil.fromimage(objetoImage): interfaz que transforma un objeto Image (de la librería PIL) en un objeto array bidimensional (de la librería Numerical Python).
- pilutil.imagesave(nombreArchivo,objetoArray): interfaz que transforma un objeto array bidimensional (de la librería Numerical Python) en un archivo de imagen.

La vectorización de un array bidimensional se ejecuta utilizando el método **reshape(nvadimension)** incluido en el objeto array.

Para ilustrar mejor como se han empleado estos métodos, se presenta a continuación un fragmento de código que lee un archivo de imagen, lo transforma en un objeto array para posteriormente vectorizarlo y poder efectuar operaciones matemáticas propias del algebra lineal:

```
#modulo PIL
import Image

#modulo SCIPy
import scipy.misc.pilutil as pilutil

#modulo NumPy
import numpy

# aperture del archive de imagen
foto  = Image.open("foto13592.jpg")

# conversión de objeto imagen a objeto array
matImg = pilutil.fromimage(foto)
# se redimensiona a un vector columna (Nx1)
vecImg = matImg.reshape((matImg.size,1))

# codigo de algebra lineal operando sobre el vector
...

# redimensionando el vector columna a objeto array
# de dimensiones 640x240
```

```
matImg = vecImg.reshape((640,240))

# almacenamiento en formato tiff
Pilutil.imsave("foto13592_Eigen.tif",matImg)
```

### *11.2 Histograma de la imagen*

El histograma de una imagen proporciona información estadística útil para determinar la calidad de una imagen; los valores a considerar son los siguientes:

- Intensidad promedio de los píxeles (brillo de la imagen)
- Distribución de los píxeles (contraste de la imagen)
- Visualización gráfica de la intensidad de los píxeles.

Un histograma ayuda a evaluar la intensidad global de la imagen, lo cual es un parámetro que influye en procesos posteriores de extracción de características de la imagen; también es importante medir como es la distribución de los píxeles así como los valores máximos y mínimos; generalmente se busca que exista una uniformidad en la distribución de píxeles, de lo contrario es probable que la imagen se encuentre contaminada con ruido lo cual dificultará la extracción de información a partir de una imagen.

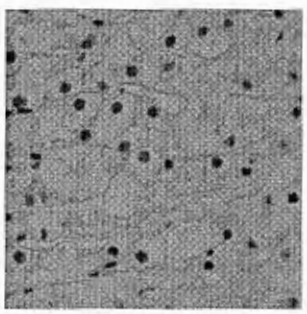


Imagen ideal para extraer información

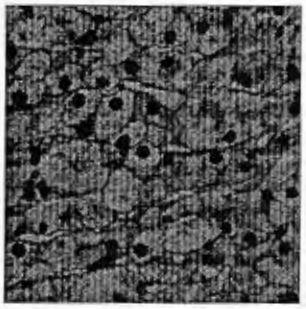
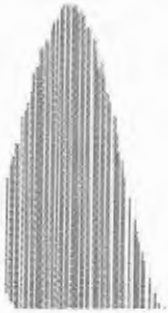


Imagen anterior de mala calidad (demasiado oscura y con ruido)

Histograma uniforme, donde la cresta representa la característica que se extraerá



Histograma no uniforme, debido a la calidad de la imagen.

*Figura 24: Ejemplo de histograma de una imagen*

Un histograma puede tener múltiples crestas; lo cual puede relacionarse a la información que se esta identificando; por ejemplo, en una imagen tomada con un microscopio electrónico, probablemente existan 2 o más crestas dependiendo de la clase de células que se estén enfocando; también una cresta puede referirse al fondo de la imagen (si el fondo no ha sido binarizado y transformado a blanco o blanco-gris).

La mejora de las imágenes se define como la técnica de procesamiento la cual incrementa el contraste de una imagen en un rango determinado de intensidad.

La definición de contraste es un poco vaga debido a la utilización en diferentes campos como son la psicología, la óptica y la fotografía; matemáticamente el contraste de una imagen en un píxel es equivalente a la diferencia entre su valor de intensidad $I(x,y)$ y la intensidad promedio $\bar{I}$ normalizada con el rango completo de intensidad:

$$C(x,y) = \frac{I(x,y) - \bar{I}}{I_{\max} - 0}$$

La formula anterior concluye que la forma mas directa de mejorar una imagen es variando el contraste expandiendo o contrayendo el rango de la intensidad de cada píxel, sin alterar la información de la imagen (únicamente se altera el histograma de la imagen).

En la práctica, algunas transformaciones de contraste modifican la intensidad de algunos píxeles de tal forma que la información de la imagen se ve reducida; ya que la transformación de contraste modifica de forma global todos los píxeles de la imagen; un mecanismo mas adecuado es utilizar métodos locales es decir el ajueste de algun tipo de filtro en ventanas de la imagen.

### 11.2.1. Tipos de histogramas

Existen tres tipos diferentes de histogramas mencionados a continuación:

#### *11.2.1.1. Histogramas de expansión ($H_E$)*

Los histogramas de expansión son denotados por $H_E$, es una transformación lineal con mucho impacto en la imagen con la característica que no afecta la información de la misma.

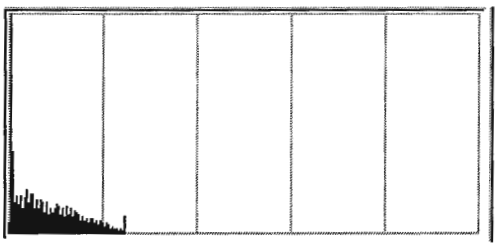
Esta transformación consiste en ajustar la intensidad de un rango, $x \in [x_{\min}, x_{\max}]$ a una intensidad que parte desde 0 hasta un límite establecido, es decir: $y \in [0, y_{\max}]$ donde típicamente $y_{\max} = 255$.

Como resultado de esta transformación el intervalo $[x_{\min}, x_{\max}]$ es ajustado de forma proporcional al nuevo y mayor intervalo $[0, y_{\max}]$; la intensidad de cada valor $x$ es transformado de acuerdo a la siguiente función linear:

$$y = \frac{x - x_{\min}}{x_{\max} - x_{\min}} y_{\max}$$

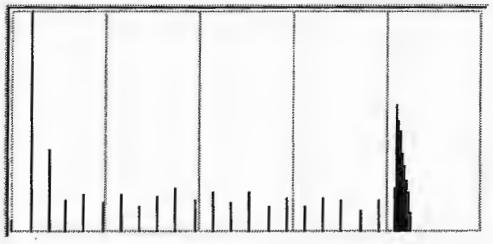
$H_E$ incrementa el contraste sin modificar la forma original del histograma:

Imagen Original



Histograma original

Imagen modificada

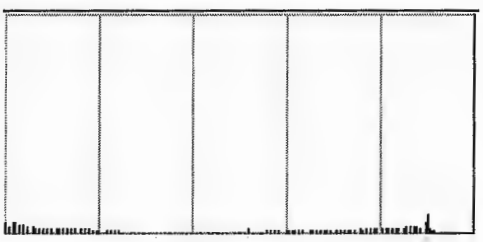


Histograma extendido

Imagen original

Histograma original

Imagen modificada



Histograma extendido

Algunas imágenes poseen poco contraste pero su histograma se expande entre los extremos (existen píxeles con intensidad 0 y 255); en este caso el Histograma extendido no será muy efectivo para mejorar la

calidad de la imagen; en este caso se utiliza un porcentaje *p* para establecer una proporción de ajuste en la intensidad y así comprimir los píxeles, es decir *p* sirve para determinar los nuevos limites inferiores y superiores $x'_{min}$, $x'_{max}$; el calculo de estas rangos se establece de la siguiente forma:

$$x'_{min} : \frac{(100-p)}{2} = \frac{100}{T} \sum_{x=x_{min}}^{x'_{min}} h(x), \quad donde \quad T = \sum_{x_{min}}^{x_{max}} h(x)$$

$$x'_{max} : \frac{(100-p)}{2} = \frac{100}{T} \sum_{x=x'_{max}}^{x_{max}} h(x)$$

De esta forma el Histograma extendido comprime los extremos de la imagen para expandir el rango del histograma.

*11.2.1.2. Histograma de salida especificada ($H_o$)*

Este tipo de histograma produce un histograma que posee la forma de una rampa linear; si el factor es positivo el rango ocupado por los píxeles de baja intensidad es ajustado y el contraste es mejorado en esa sección; si el factor es negativo el rango utilizado por los píxeles de alta intensidad es ajustado y mejorado. Este tipo de histograma es muy útil para efectuar ajustes de contraste de forma automática.

Uno de los requisitos para este algoritmo es calcula la suma parcial de los píxeles la cual viene determinada por la siguiente formula:

$T(x) = \sum_{x_{min}}^{x} h(x) - \frac{1}{2}[h(x_{min}) + h(x_{max})]$, donde $h(x)$ es la intensidad de un píxel determinado, $x_{min}$ y $x_{max}$ es la posición de los píxeles con menor y mayor intensidad.

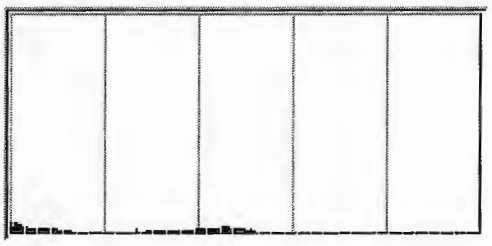
La transformación lineal se especifica de la siguiente manera:

$$y = \begin{cases} \frac{y_{max}}{2} - \frac{T_{max}}{m y_{max}} - \sqrt{\left(\frac{T_{max}}{m y_{max}}\right)^2 + \frac{2T(x)}{m}} & , \quad m < 0 \\ \frac{y_{max}^2}{T_{max}} T(x) & , \quad m = 0 \\ \frac{y_{max}}{2} - \frac{T_{max}}{m y_{max}} + \sqrt{\left(\frac{T_{max}}{m y_{max}} - \frac{y_{max}}{2}\right)^2 + \frac{2T(x)}{m}} & , \quad m > 0 \end{cases}$$

La experiencia sugiere que el factor $m$ a calcular depende de la intensidad de la imagen según las siguientes relaciones:

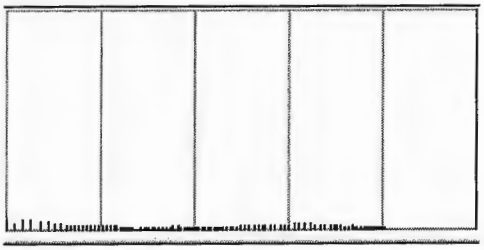
- Imágenes con mucha intensidad: $m = \frac{-2T}{y_{max}^2}$
- Imágenes con poca intensidad: $m = \frac{2T}{y_{max}^2}$
- Para ecualizar el histograma: $m = 0$

Imagen original



Histograma original

Imagen modificada



Histograma de salida

*11.2.1.3. Histograma de Transformación especificada ($H_T$).*

La tercera clase de histograma de transformación, $H_T$, requiere explícitamente especificaciones para efectuar las transformaciones, por lo cual es muy utilizado cuando el tratamiento de una imagen se realiza a través de una inspección visual interactiva.

Para aplicar esta técnica, la función de transformación requiere un rango de intensidades de entrada y un rango más amplio de intensidades de salida; de la siguiente forma:

Sea $x = x_c$ donde representa un valor de intensidad deseado o un promedio de intensidades, sea $m$ un valor de tal forma que $m > 1$ el cual controlará el grado de transformación en la imagen, se tiene un rango de intensidades de entrada $x \in [x_{\min}, x_{\max}]$ y un rango de salida $y \in [0, y_{\max}]$ el cual es más amplio; la intensidad de salida se define como el siguiente polinomio:

$$y_s = Ax^3 + Bx^2 + Cx + D$$

Donde los coeficientes vienen dados por las siguientes formulas:

$$A = \frac{1-m}{x_{\max}^2 - 3x_c x_{\max} + 3x_c^2}$$
$$B = -3Ax_c$$
$$C = m + 3Ax_c^2$$
$$D = 0$$

Por diseño, $y(x)$ debe estar limitado la rango de intensidades de salida $[0, y_s]$ sin embargo pudiera darse el caso en que un valor de entrada ocasione que se exceda $y_s$, en estos casos se debe fijar el valor de la intensidad de salida para que se respete los rangos de salida; sin embargo, si se dan muchos casos de estos la imagen tiende a deformarse por lo que la mejor técnica para forzar a que se cumpla que la salida se encuentre entre los rangos especificados es utilizar una función con forma de rampa de la siguiente forma:

Limite inferior: $y_{lower} = kx$

Limite superior: $y_{upper} = k(x - x_{\max}) + y_{\max}$

Donde *k* es un parámetro adicional ajustable que representa la pendiente de la rampa, considerando que si *k* se aproxima a 0 la pendiente se disminuye y si *k* se aproxima a 1 la pendiente se pronuncia de tal forma que la imagen se deforma demasiado; un valor típico de *k* es *k=0.2* .

La función del Histograma de Transformación queda definida finalmente de la siguiente forma:

$$y = \begin{cases} y_{lower}(x), & y_s(x) < y_{lower}(x) \\ y_{upper}(x), & y_s(x) > y_{upper}(x) \\ y_s(x), & default \end{cases}$$

## *11.3. Operaciones Locales: La Convolución*

La operación de convolución es fundamental para el análisis de las imágenes, el proceso calcula un nuevo valor de un píxel basándose en la evaluación de un promedio ponderado de los píxeles vecinos en una región de *kxk* píxeles donde el píxel a modificar se encuentra en el centro.

Los pesos o valores con los que se pondera el promedio de intensidad de los píxeles vecinos son conocidos como mascaras, filtros o kernels; y están compuestos por una matriz cuadrada de valores.

La operación de la convolución es define a través de la siguiente formula:

$$g_o(x,y) = \sum_{\xi=-\infty}^{\infty} \sum_{\eta=-\infty}^{\infty} g_i(\xi,\eta) h(x-\xi, y-\eta)$$

Donde $x, y, \xi, \eta$ son enteros.

Normalmente, los kernels que se emplean son regiones cuadradas (aunque pueden existir kernels rectangulares), de tal forma que $k_x = k_y \equiv k$, donde $k$ es un numero par mucho mas pequeño que la dimensión de la imagen, la formula de la convolución puede escribirse mas simplificada de la siguiente forma:

$$g_o(x,y) = \sum_{\xi=-(k-1)/2}^{(k-1)/2} \sum_{\eta=-(k-1)/2}^{(k-1)/2} g_i(\xi,\eta) h(x-\xi, y-\eta)$$

Consideraciones especiales:

a. Bordes: La implementación de la convolución de acuerdo a la formula original inevitablemente corrompe los píxeles del borde, debido a que la mascara se aplica sobre píxeles inexistentes; esto no es un serio problema en la practica ya que la información próxima a los bordes de la imagen normalmente no es muy relevante; sin embargo cuando el tamaño del kernel es muy amplio (*k* es grande) se pueden corromper datos próximos al borde. Las mejores prácticas para el tratamiento de bordes son las siguientes:

   - Los píxeles en las 4 esquinas de los bordes son fijados a 0 y es ejecutado el proceso de la convolución en el remanente de la imagen de tamaño *N-(k-1)/2*.

   - Las filas y columnas son consideradas son consideradas como cubiertas por una mascara de 5x5 cuando el kernel es de 3x3, esta técnica tiene la ventaja de una fácil implementación y recibe el nombre de convolución cíclica; el resultado es un borde menos corrompido.

b. Separabilidad: La aplicación de los kernels pueden realizarse de forma separada tal como se ilustra en la siguiente formula:

$$h(x-\xi, y-\eta) = h(x-\xi)h(y-\eta)$$

La aplicación de kernels separados en 2 dimensiones se realizan a través de 2 operaciones en 1 dimensión, esto es muy frecuente en la práctica, por ejemplo el Filtro Gausiano (low-pass) es separable, así como otros filtros para detección de bordes, etc.

Por ejemplo, considere el siguiente kernel 3x3 Gausiano:

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix} = \frac{1}{4}\begin{bmatrix}1\\ 2\\ 1\end{bmatrix} \frac{1}{4}\begin{bmatrix}1 & 2 & 4\end{bmatrix}$$

Se puede observar que la aplicación de la mascara se puede realizar en 2 pasos o ejecutar la matriz factorizada, lo cual es mucho mas eficiente en términos computacionales.

## *11.4. Reducción de Ruido (kernels uniformes)*

La operación de suavizado de las imágenes se realiza a través del promedio de los píxeles en la proximidad; se utiliza comúnmente cuando existe mucha variación en la imagen usualmente relacionada con ruido; el algoritmo es el siguiente: cada píxel de la imagen será reemplazado por un promedio de los valores de los píxeles próximos en el rango de *kxk*.

El kernel se define de la siguiente manera:

$$h(x-\xi, y-\eta) = \frac{1}{k^2} \quad para \quad -\frac{(k-1)}{2} \le \xi, \eta \le \frac{(k-1)}{2},$$

Donde usualmente se selecciona un valor de k par, de esta forma la suma total de las intensidades de la región kxk es divida entre el total de píxeles en dicha región.

El tamaño del parámetro *k* viene dado por la función $k = 2w_n + 1$ *si las impurezas son de tamaño pequeño y* $k \leq 2w_n - 1$, donde $w_n$ es el tamaño de las impurezas que se desean eliminar; por ejemplo si las impurezas del ruido no superan los 2 píxeles se recomendara una kernel de dimensiones: 2(2)+1=5; mientras que si las impurezas poseen un diámetro de 4 píxeles o mas, se recomienda un kernel de tamaño máximo: 2(4)-1=7 ya que un kernel demasiado grande puede ocasionar perdida de información en la imagen.

Los efectos de aplicar dicho kernel se reflejan en las siguientes figuras:

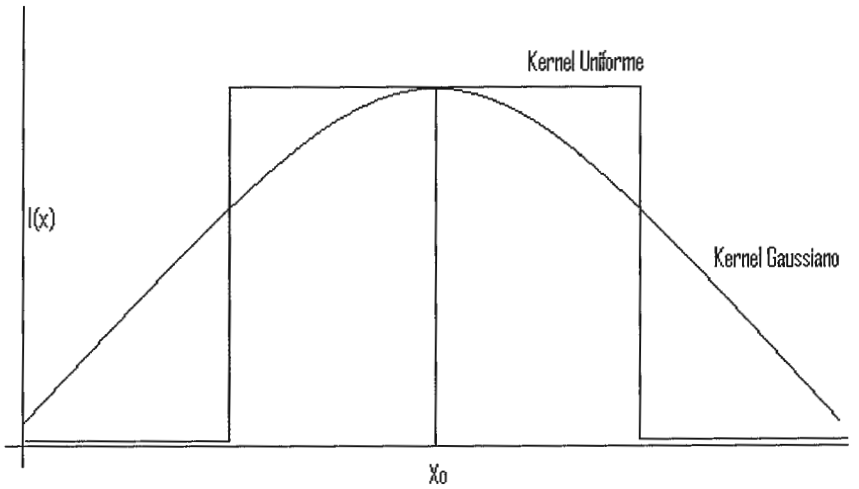
Imagen Original

Imagen suavizada

Es importante recalcar que un kernel de suavizado reduce, no elimina el ruido presente en una imagen, además la información reelevante también se ve afectada por este tipo de kernels.

## *11.5. Kernel Gaussiano*

El kernel Gaussiano tiene la forma acampanada perfecta, parecido a una parábola, donde el punto más alto se ubica en el centro de la figura.




La aplicación del kernel Gaussiano produce en cada píxel de la imagen un promedio ponderado de tal forma que los píxeles centrales contribuyen en mayor proporción que los píxeles de los extremos.

Los coeficientes del kernel Gaussiano se obtienen a través de la siguiente formula:

$h_G(i,j) = \exp\left[-1/2(d/\sigma)^2\right]$, donde $d = (i^2 + j^2)^{1/2}$, $-(k-1)/2 \le i,j \le (k-1)/2$

El parámetro de la desviación típica $\sigma$ es determinado por el diseñador del algoritmo; mientras más alto sea su valor incrementa la agresividad de la reducción de ruido sin embargo incrementa la posibilidad de destrucción de la información en la imagen; matemáticamente un valor de $\sigma = \infty$ es equivalente a la aplicación de un kernel uniforme.

La desviación típica $\sigma$ depende también del tamaño del kernel a través de la siguiente relación: $\sigma=(2w+1)/2,\quad k>2w+1$, como k debe ser par de preferencia, es común aplicar las siguientes estimaciones: $k=2w+3$ ó $k=2w+5$.

A continuación se muestran ejemplos de kernels Gaussianos de diferentes dimensiones:

*B1* *1/2(1 1)*

*B2* *1/4(1 2 3)*

*B3* *1/8(1 3 3 1)*

*B4* *1/16(1 4 6 4 1)*

*B5* *1/32(1 5 10 10 5 1)*

*B6* *1/64(1 6 15 20 15 6 1)*

*B7* *1/128(1 7 21 35 35 21 7 1)*

*B8* *1/256(1 8 28 56 70 56 28 8 1)*

Donde el kernel se forma de la siguiente forma:

$$B2=\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}=\frac{1}{4}\begin{bmatrix}1\\ 2\\ 1\end{bmatrix}\frac{1}{4}\begin{bmatrix}1 & 2 & 1\end{bmatrix}$$

$$B4=\frac{1}{256}\begin{bmatrix}1 & 4 & 6 & 4 & 1\\ 4 & 16 & 24 & 16 & 4\\ 6 & 24 & 36 & 24 & 6\\ 4 & 16 & 24 & 16 & 4\\ 1 & 4 & 6 & 4 & 1\end{bmatrix}$$

Se puede observar que el tamaño del kernel incrementa la cantidad de operaciones computacionales necesarias para la aplicación.

# 12. Diseño del prototipo

## 12.1. Casos de uso del prototipo

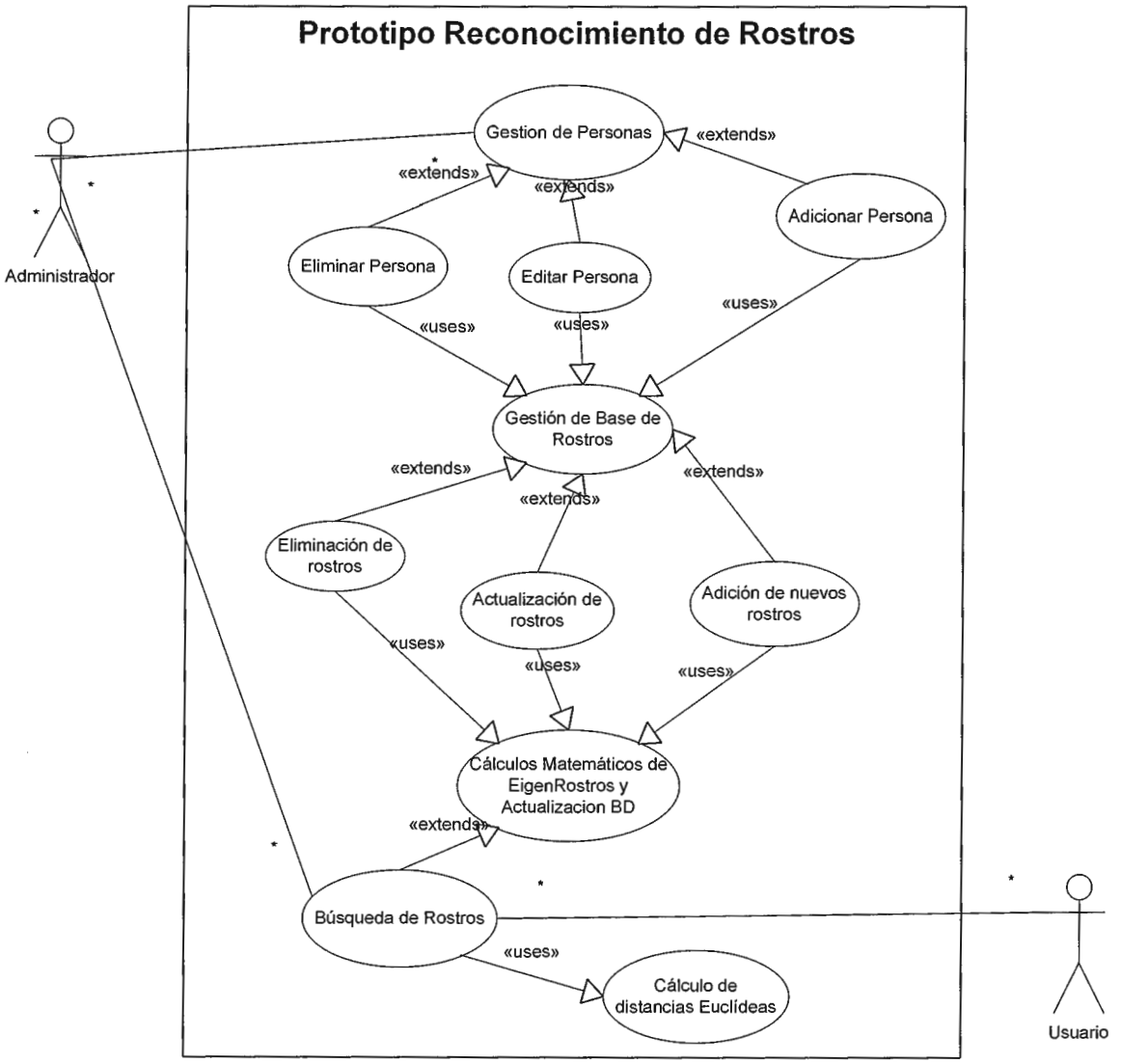



*Figura 25: Diagrama de Casos de Uso*

Existen 2 actores principales en el proceso de un sistema biométrico de reconocimiento de rostros:

- Administrador: Usuario responsable de la gestión de la base de datos de rostros, lo cual implica actualización de información, eliminación de registros y entrenamiento de la base conforme nuevos rostros sean agregados al sistema.
- Usuario: usuario con permisos limitados, la única opción que posee es la búsqueda de rostros y visualización de información.

### 12.1.1. Descripción de caso de uso: Búsqueda de Rostros

| | |
|---|---|
| **Nombre:** | Búsqueda de Rostros |
| **Objetivos:** | Realizar una búsqueda en la base de datos a partir de una fotografía digitalizada en un formato de archivo PNG, JPG, TIFF o BMP. |
| **Actores:** | Usuario, Administrador |
| **Precondiciones:** | - Base de rostros actualizado |
| **Garantías de éxito (post-condiciones) :** | Visualización de un conjunto de rostros con cierto grado de similitud con la imagen desconocida. |
| **Escenario principal de éxito:** | 1 Usuario selecciona una imagen digital correspondiente a un rostro desconocido.<br>2 El sistema calcula las proyecciones de la imagen sobre los eigenrostros representativos.<br>3 El módulo Cálculo de distancias totaliza las diferencias entre las proyecciones de la imagen desconocida y las proyecciones de las imágenes de la Base de Rostros.<br>4 Se seleccionan aquellos valores inferiores a un *threshold* y se reconstruyen las imágenes basándose en esos valores. |
| **Extensiones (flujos alternativos):** | 1 El usuario selecciona una imagen que no cumple con los requisitos de formato de archivo.<br>1.1 El sistema envía mensaje de "Formato de archivo irreconocible".<br>2 El usuario selecciona una imagen que cumple con los |

| | |
|---|---|
| | requisitos de formato de archivo (JPEG, PNG, BMP o TIFF) pero no cumple con la resolución que poseen la Base de Rostros.<br>2.1 El sistema envía mensaje de: "Resolución no coincide con imágenes en Base de Rostros". |
| **Requisitos:** | - Imagen digitalizada del rostro desconocido debe ser de la misma resolución que los rostros almacenados en el sistema biométrico.<br>- Imagen debe estar digitalizada en un formato válido (JPEG, PNG, BMP o TIFF) |
| **Resultados:** | Conjunto de imágenes reconstruidas a partir de los eigenrostros y un % de similitud con la imagen original digitalizada. |

## 12.1.2. Descripción de caso de uso: Cálculo de Distancias Euclideas.

| | |
|---|---|
| **Nombre:** | Cálculo de Distancias Euclideas |
| **Objetivos**: | Calcular la diferencia entre las proyecciones de una imagen desconocida y los eigenrostros representativos vs las proyecciones originales de las imágenes y los eigenrostros representativos. |
| **Actores**: | Administrador, Usuario |
| **Precondiciones**: | - Proyecciones de la imagen desconocida sobre los eigenrostros representativos.<br>- Proyecciones de las imágenes de la Base de rostros sobre los eigenrostros representativos. |
| **Garantías de éxito (post-condiciones):** | Vector cuantificado de similitudes. |
| **Escenario principal de éxito:** | Calculo de diferencia entre proyecciones calculadas de la imagen desconocida y las imágenes de la base de rostros. |
| **Extensiones (flujos alternativos):** | Ninguno. |
| **Requisitos:** | - Proyecciones calculadas pertenecientes a la imagen |

| | |
|---|---|
| | desconocida.<br>- Proyecciones calculadas pertenecientes a las imágenes de la base de rostros. |
| **Resultados:** | Vector ordenado ascendentemente según la diferencia entre proyecciones. |

### 12.1.3. Descripción de caso de uso: Gestión de Personas

| | |
|---|---|
| **Nombre:** | Gestión de Personas |
| **Objetivos:** | Permitir las adición, eliminación y actualización de personas al sistema biométrico. |
| **Actores:** | Administrador |
| **Precondiciones:** | - Datos de la persona que se desea eliminar, modificar o agregar.<br>- Imagen que desea asociarse o actualizar. |
| **Garantías de éxito (post-condiciones):** | Base de datos actualizada. |
| **Escenario principal de éxito:** | Según la información que se posea, se ingresa al módulo de adicionar, eliminar o modificar personas; cada uno de esos módulos permitirá adicionalmente el acceso a la administración de rostros. |
| **Extensiones (flujos alternativos):** | Ninguno. |
| **Requisitos:** | Información persona que se desee modificar o agregar, incluyendo imagen digitalizada de rostro. |
| **Resultado** | Base de datos actualizada. |

### 12.1.4. Descripción de caso de uso: Adicionar Persona

| | |
|---|---|
| **Nombre:** | Adicionar Persona |
| **Objetivos:** | Interfaz para ingresar nuevas personas al sistema Biométrico de Rostros. |

| | |
|---|---|
| **Actores:** | Administrador |
| **Precondiciones:** | Información que se desea agregar referente a una persona:<br>- 1 ó más imágenes de rostro.<br>- Nombre 1: Primer nombre<br>- Nombre 2: Segundo nombre<br>- Apellido 1: Primer apellido<br>- Apellido 2: Segundo apellido<br>La imagen digitalizada que desea adicionarse a la base de rostros cumple con condiciones de formato, resolución, enfoque y calidad. |
| **Garantías de éxito (post-condiciones):** | Nueva persona en el sistema; la base de rostros ha sido actualizada (recálculo de proyecciones, eigenrostros, eigenvectores y eigenvalores) |
| **Escenario principal de éxito:** | 1 Usuario Ingresa los datos personales al módulo de adicionar persona.<br>2 El sistema asigna un ID único, y almacena permanentemente los datos sin rostro.<br>3 El usuario ingresa al módulo de Gestión de Base de Rostros para adicionar posteriormente el rostro de la persona. |
| **Extensiones (flujos alternativos:** | Ninguno. |
| **Requisitos:** | - La fotografía no debe estar actualmente asignada en la base de rostros.<br>- Acceso al módulo de Adicionar Persona. |
| **Resultado:** | Base de datos con un nuevo registro. |

### 12.1.5. Descripción de caso de uso: Editar Persona

| | |
|---|---|
| **Nombre:** | Editar Persona |
| **Objetivos:** | Interfaz para editar datos de las personas registradas en el sistema biométrico. |
| **Actores:** | Administrador. |

| | |
|---|---|
| **Precondiciones:** | ID de la persona que se desea modificar debe existir en el sistema biométrico. |
| **Garantías de éxito (post-condiciones):** | Base de datos actualizada, recalculo de proyecciones, eigenrostros, eigenvectores y eigenvalores. |
| **Escenario principal de éxito:** | 1 Usuario ingresa ID de la persona que desea actualizar.<br>2 Sistema biométrico ubica el registro y reconstruye las fotografías almacenadas de la persona.<br>3 Usuario edita los datos personales (nombre1, nombre2, apellido1 y apellido2)<br>4 Usuario ingresa a módulo de gestión de rostros para adicionar, modificar o eliminar rostros asociados al ID de la persona que se está actualizando. |
| **Extensiones (flujos alternativos):** | 1 Usuario ingresa ID de la persona que desea actualizar.<br>2 Sistema biométrico no ubica ID proporcionado por usuario; retorna mensaje: "ID de persona no encontrado". |
| **Requisitos:** | Acceso a módulo de Editar Personas y proporcionar un ID existente. |
| **Resultado:** | Base de datos actualizada. |

## 12.1.6. Descripción de caso de uso: Eliminar Persona.

| | |
|---|---|
| **Nombre:** | Eliminar Persona |
| **Objetivos:** | Interfaz para eliminar una persona del sistema biométrico y todos los rostros asociados a ella. |
| **Actores:** | Administrador |
| **Precondiciones:** | ID de la persona debe existir en la base de datos. |
| **Garantías de éxito (post-condiciones):** | Persona es eliminada de la base de datos, recalculo de eigenrostros, eigenvectores y eigenvalores. |
| **Escenario principal de éxito:** | 1 Usuario ingresa ID de la persona que se desea eliminar del sistema biométrico.<br>2 Sistema biométrico ubica el registro y reconstruye las fotografías almacenadas de la persona. |

| | |
|---|---|
| | 3 Se elimina el registro junto con todas las proyecciones de la base de rostros. |
| **Extensiones (flujos alternativos):** | 1 Usuario ingresa ID de la persona que se desea eliminar del sistema biométrico.<br>2 Sistema biométrico busca por ID el registro y no se encuentra en la base de datos.<br>3 Sistema retorna mensaje de error: "ID de persona no encontrado". |
| **Requisitos:** | - Acceso a modulo de eliminar personas.<br>- ID de registro que desea eliminarse. |
| **Resultado:** | Base de datos actualizada. |

## 12.1.7. Descripción de caso de uso: Gestión de Rostros

| | |
|---|---|
| **Nombre:** | Gestión de Rostros |
| **Objetivos:** | Interfaz para adicionar, modificar y eliminar rostros del sistema Biométrico. |
| **Actores:** | Administrador |
| **Precondiciones:** | - Información que desea modificarse (imagen digital de un rostro presente en la base de rostros, datos personales), adicionarse (imagen digital de un rostro nuevo no existente en la base de datos, datos personales) o eliminarse (ID de la persona que desea eliminarse).<br>- La imagen digitalizada que desea adicionarse a la base de rostros cumple con condiciones de formato, resolución, enfoque y calidad.<br>- El ID del rostro que desea eliminarse es válido (existe previamente en la base de datos) |
| **Garantías de éxito (post-condiciones):** | Actualización de Base de Rostros |
| **Escenario principal de éxito:** | Usuario ingresa a módulo de gestión de rostros después de haber procesado exitosamente la adición/modificación de un |

| | |
|---|---|
| | rostro en el módulo de gestión de personas. |
| **Extensiones (flujos alternativos):** | Ninguno. |
| **Requisitos:** | - Acceso al módulo de Gestión de la Base de Rostros, el cual utiliza internamente los módulos de Eliminación de Rostros, Actualización de Rostros, Adición de Rostros.<br>- Acceso al módulo de Búsqueda de Rostros para evitar duplicidad de información.<br>- Acceso al módulo para Recálculo de Eigenrostros. |
| **Resultados:** | Actualización en la Base de Datos de Rostros; dicha actualización se reflejará en la siguiente búsqueda de rostros. |

### 12.1.8. Descripción de caso de uso: Adición de nuevos rostros

| | |
|---|---|
| **Nombre:** | Adición de nuevos rostros |
| **Objetivos:** | Asociar nuevos rostros a una persona ya registrada en el sistema biométrico. |
| **Actores:** | Administrador |
| **Precondiciones:** | - ID de la persona debe de existir en el sistema biométrico.<br>- El nuevo rostro debe cumplir características de resolución, claridad de imagen, formato de archivo (JPEG, PNG, TIFF o BMP) |
| **Garantías de éxito (post-condiciones):** | Nuevo rostro asociado a una persona existente en el sistema biométrico. |
| **Escenario principal de éxito:** | 1 Usuario ingresa ID de la persona a la que desea asociarle un rostro.<br>2 Sistema localiza la persona, reconstruye los rostros asociados.<br>3 Usuario carga en sistema la nueva imagen de rostro.<br>4 Sistema busca nueva imagen en la base de rostros existente para evitar una duplicidad de rostros; si la imagen no es localizada se procede a calcular proyecciones, calcular eigenvectores, eigenvalores y |

| | |
|---|---|
| | eigenrostros.<br>5 Se actualiza base de rostros. |
| **Extensiones (flujos alternativos):** | 1 Usuario ingresa ID de la persona a la que desea asociarle un rostro.<br>1a Sistema intenta localizar ID pero falla por error en código de persona; sistema retorna mensaje: "ID de persona incorrecto".<br>2 Usuario ingresa ID de la persona a la que desea asociarle un rostro.<br>2a Sistema localiza registro de la persona, presenta datos personales y una reconstrucción de los rostros almacenados en la base.<br>2b Usuario selecciona nuevo rostro que desea asociar a la persona, dicho rostro no se encuentra digitalizado en un tipo de formato válido (JPEG, BMP, TIFF o PNG).<br>2c Sistema retorna mensaje de error: "Tipo de archivo de imagen no soportado".<br>3 Usuario ingresa ID de la persona a la que desea asociarle un rostro.<br>3a Sistema localiza registro de la persona, presenta datos personales y una reconstrucción de rostros almacenados en la base<br>3b Usuario selecciona nuevo rostro; dicha imagen se encuentra digitalizada en un formato válido pero no cumple con las características de solución de los rostros almacenados inicialmente.<br>3c Sistema retorna mensaje: "Resolución de la imagen no coincide con las almacenadas". |
| **Requisitos:** | - ID de registro de persona existente<br>- Acceso a módulo de gestión de personas y adición de nuevo rostro. |
| **Resultado:** | Nuevo rostro asociado a una persona existente en el sistema. |

## 12.1.9. Descripción de caso de uso: Actualización de Rostro

| | |
|---|---|
| **Nombre:** | Actualización de Rostro |
| **Objetivos:** | Interfaz para actualizar una imagen existente en la base de rostros. |
| **Actores:** | Administrador. |
| **Precondiciones:** | - ID de la persona debe existir en el sistema biométrico. |
| **Garantías de éxito (post-condiciones):** | Base de rostros actualizada con nueva imagen. |
| **Escenario principal de éxito:** | 1 Usuario ingresa ID de persona a la que desea modificar el rostro.<br>2 Sistema localiza persona y reconstruye rostros asociados.<br>3 Usuario selecciona rostro reconstruido que desea modificar e indica ubicación de archivo de imagen que será el reemplazo.<br>4 Sistema busca archivo de imagen para evitar duplicidad de rostros; si no encuentra la imagen en la base de rostros existente procede a calcular proyecciones, eigenrostros, eigenvectores y eigenvalores.<br>5 Actualización de la base de rostros. |
| **Extensiones (flujos alternativos):** | 1 Usuario ingresa ID de persona a la que desea modificar el rostro.<br>1a Sistema intenta localizar persona, pero el ID es incorrecto, el sistema retorna mensaje: "ID de persona inválido".<br>2 Usuario ingresa ID de persona a la que desea modificar el rostro.<br>2a Sistema localiza a la persona basándose en el ID y reconstruye las imágenes que tiene asociadas.<br>2b Usuario selecciona que imagen (de la base de rostros) va a reemplazar.<br>2c Usuario selecciona el nuevo archivo de imagen que actualizará, sin embargo este archivo no se encuentra en un formato soportado (JPEG, BMP, |

| | |
|---|---|
| | TIFF, PNG)<br>2d Sistema retorna mensaje de error: "Tipo de imagen no soportado".<br>3 Usuario ingresa ID de persona a la que desea modificar el rostro.<br>3a Sistema localiza a la persona basándose en el ID y reconstruye las imágenes que tiene asociadas.<br>3b El Actor selecciona el nuevo archivo de imagen que actualizará, este archivo es un formato válido.<br>3c El sistema efectúa una búsqueda del archivo, y encuentra demasiada similitud de la imagen con alguna existente en la base de rostros.<br>3d El sistema retorna mensaje: "Imagen ya existente en base de rostros". |
| **Requisitos:** | - ID de persona registrada en sistema biométrico.<br>- Imagen, en formato válido, que desea asociarse al ID de la persona. |
| **Resultado:** | Base de rostros actualizada con nueva imagen. |

## 12.1.10. Descripción de caso de uso: Eliminación de rostros

| | |
|---|---|
| **Nombre:** | Eliminación de rostros |
| **Objetivos:** | Interfaz para des-asociar rostros de las personas registradas en el sistema biométrico. |
| **Actores:** | Administrador |
| **Precondiciones:** | - ID de la persona que se desea modificar. |
| **Garantía de éxito (post-condiciones):** | Modificación de rostros asociados con la persona. |
| **Escenario principal de éxito:** | 1 Actor ingresa ID de persona registrada en sistema biométrico.<br>2 Sistema localiza a la persona y reconstruye rostros asociados.<br>3 Actor selecciona rostro que desea eliminar |

| | |
|---|---|
| | 4 Sistema elimina proyecciones, recalcula eigenrostros, eigenvectores y eigenvalores.<br>5 Actualización en base de rostros. |
| **Extensiones (flujos alternativos):** | 1 Actor ingresa ID de persona.<br>2 Sistema falla en localizar ID de la persona (no esta registrada).<br>3 Sistema retorna mensaje: "ID de persona no encontrada" |
| **Requisitos:** | - ID de persona registrada<br>- Acceso a módulo de gestión de personas y gestión de rostros. |
| **Resultado:** | Base de rostros actualizada. |

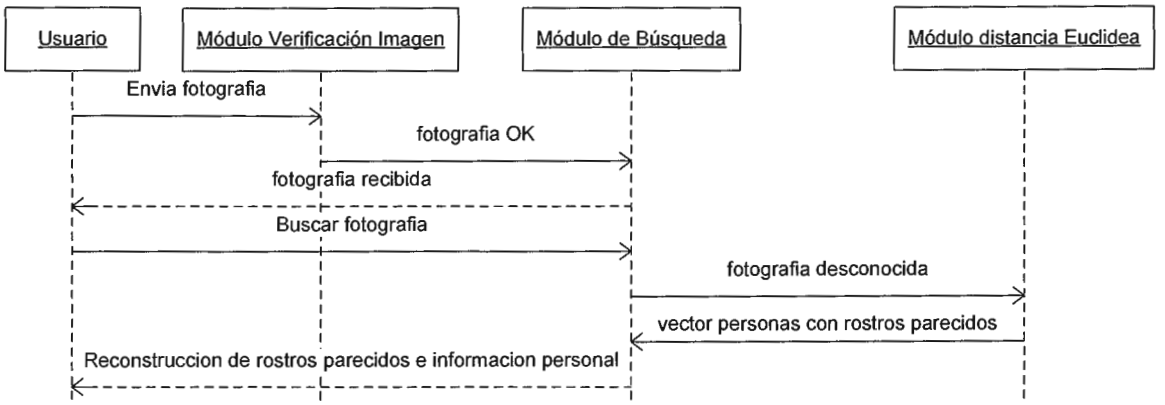
### 12.1.11. Descripción de caso de uso: Cálculos de EigenRostros

| | |
|---|---|
| **Nombre:** | Cálculos de EigenRostros |
| **Objetivos:** | Interfaz para recalcular eigenrostros y actualizar base de rostros. |
| **Actores:** | Administrador |
| **Precondiciones:** | Registro actual de personas (datos personales y fotografía original) a las que se les calcularán los eigenrostros. |
| **Garantía de éxito (post-condiciones):** | Base de Rostros recalculada. |
| **Escenario principal de éxito:** | 1 Actor ingresa a la opción de cálculo de EigenRostros<br>2 Sistema solicita confirmación para recalcular toda la base de rostros basándose en las fotografías existentes asociadas a cada persona en el sistema biométrico.<br>3 Actor confirma recálculo de eigenrostros y por consecuencia reconstrucción de base de rostros. |
| **Extensiones (flujos alternativos):** | Ninguno |
| **Requisitos:** | - Acceso a módulo de Cálculo de Eigenrostros.<br>- Registro de personas existente en sistema biométrico. |
| **Resultado:** | Base de rostros recalculada. |

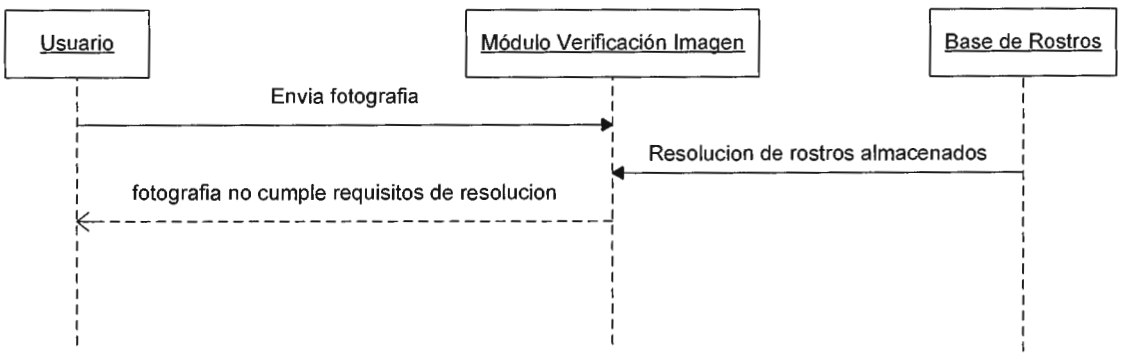
## *12.2. Diagramas de Secuencia*

### 12.2.1. Diagrama de Secuencia: Búsqueda de Rostros.

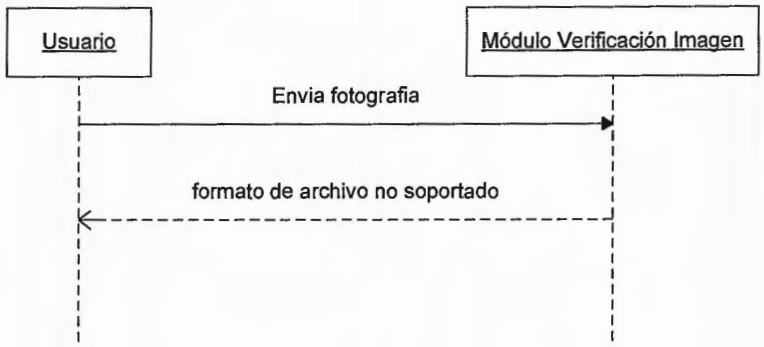
#### *12.2.1.1. Escenario principal de éxito.*




#### *12.2.1.2. Escenario alternativo 1.*

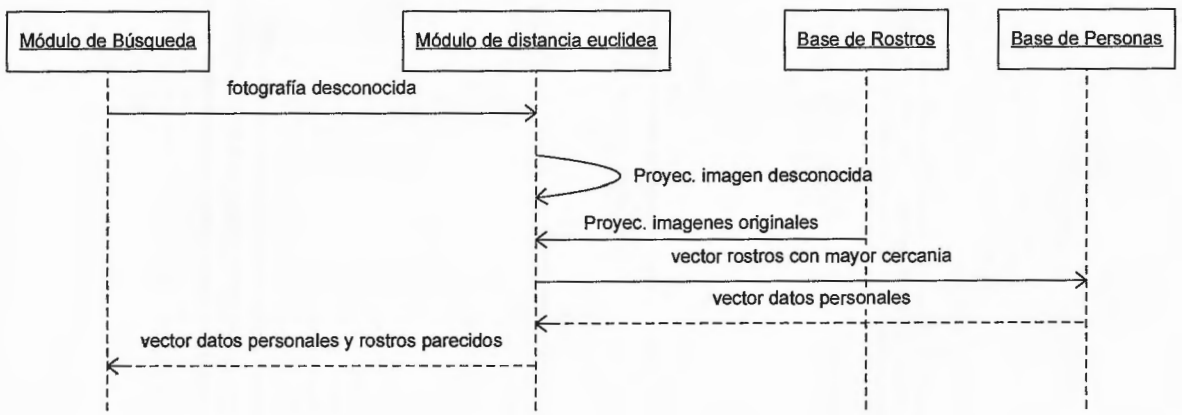


### *12.2.1.3. Escenario alternativo 2.*



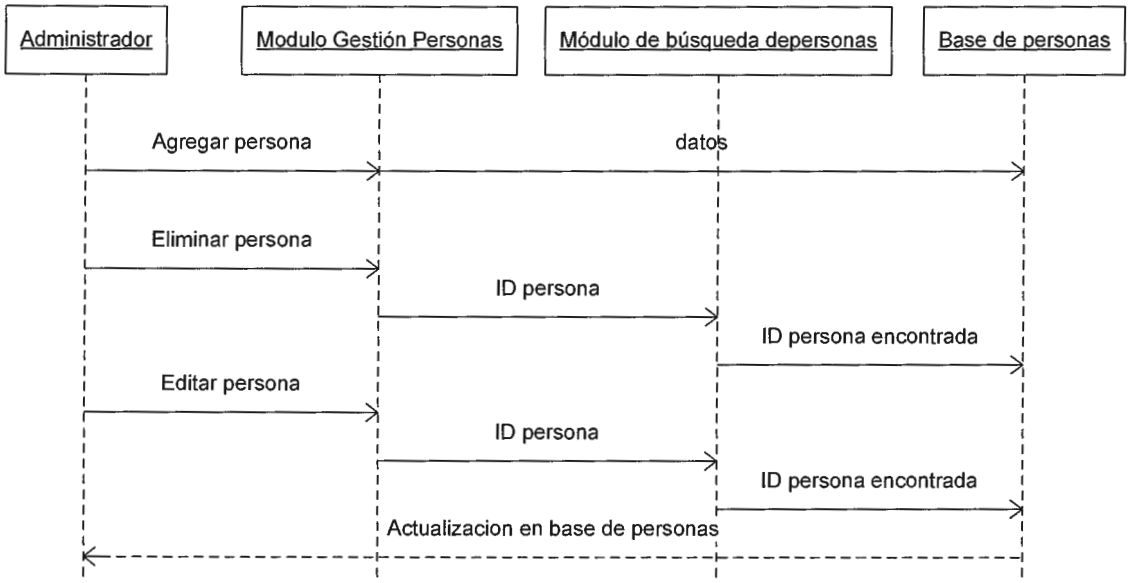

## 12.2.2. Diagrama de Secuencia: Cálculo de distancias euclideas.

### *12.2.2.1. Escenario principal de éxito.*

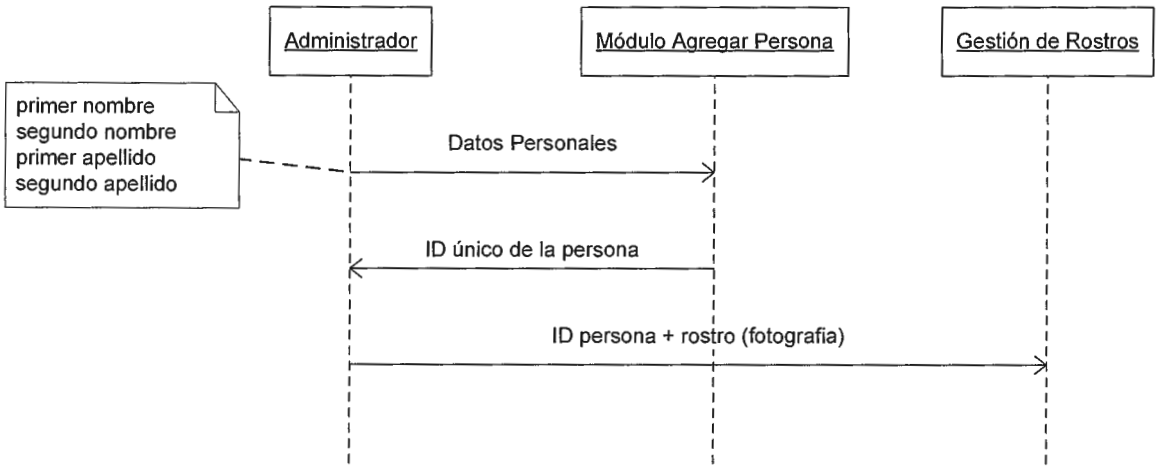


## 12.2.3. Diagrama de Secuencia: Gestión de Personas

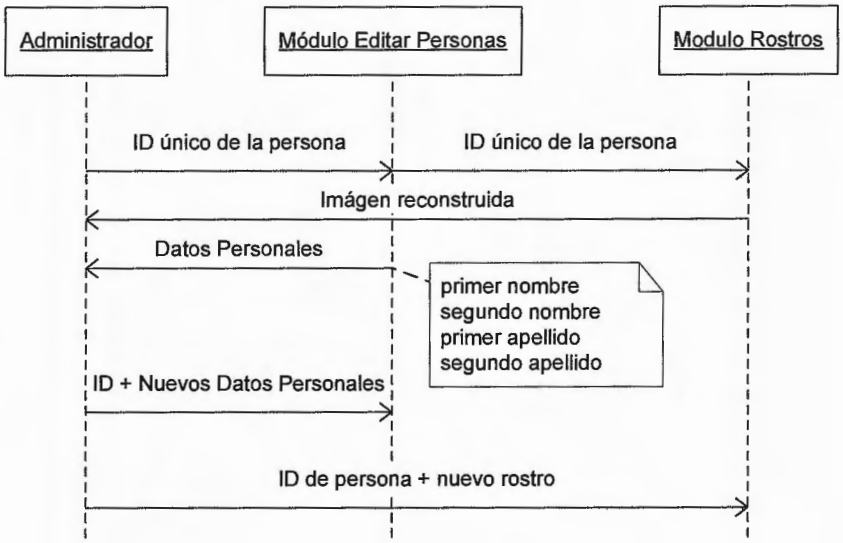
### *12.2.3.1. Escenario principal de éxito.*



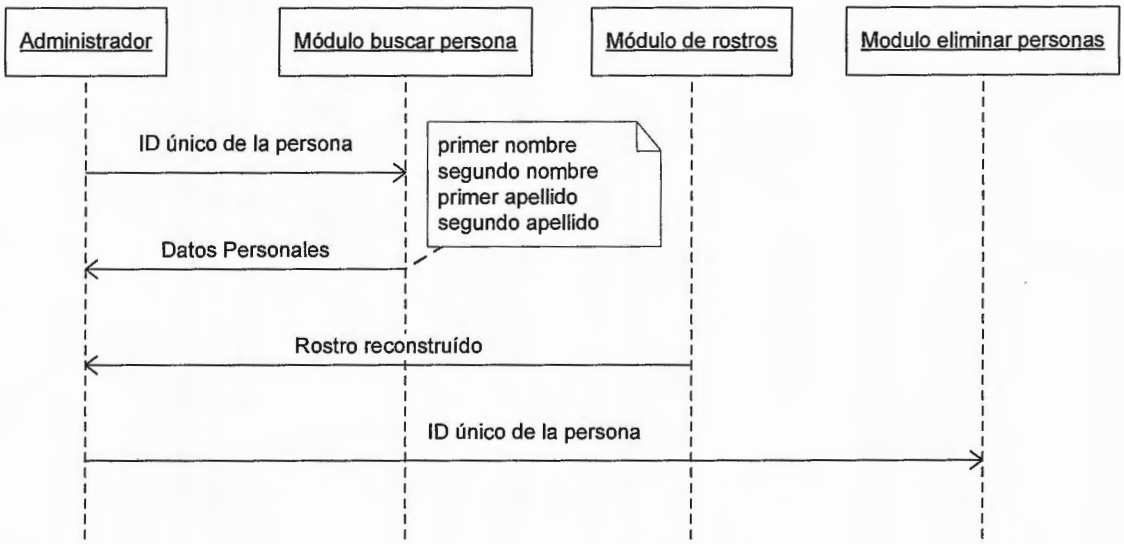

### *12.2.3.2 Escenario Adicionar Persona (Principal)*



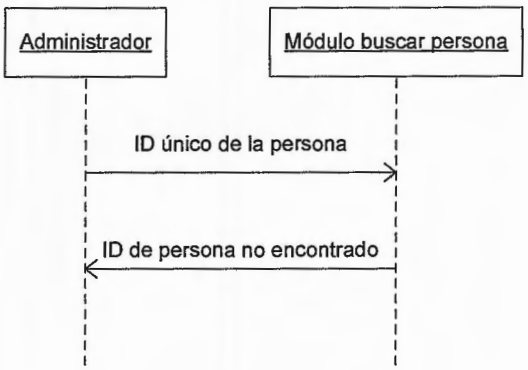
### *12.2.3.3 Escenario Editar Persona (Principal)*



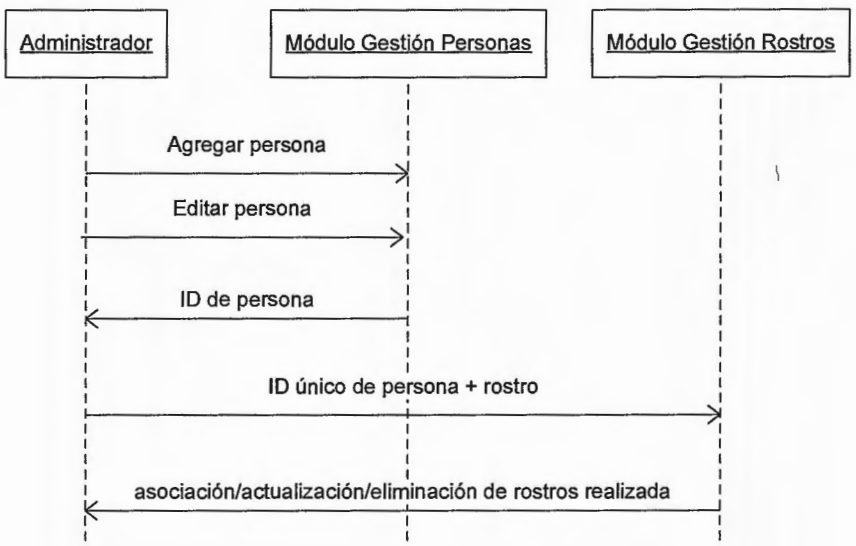

### *12.2.3.4 Escenario Eliminar Persona (Principal)*



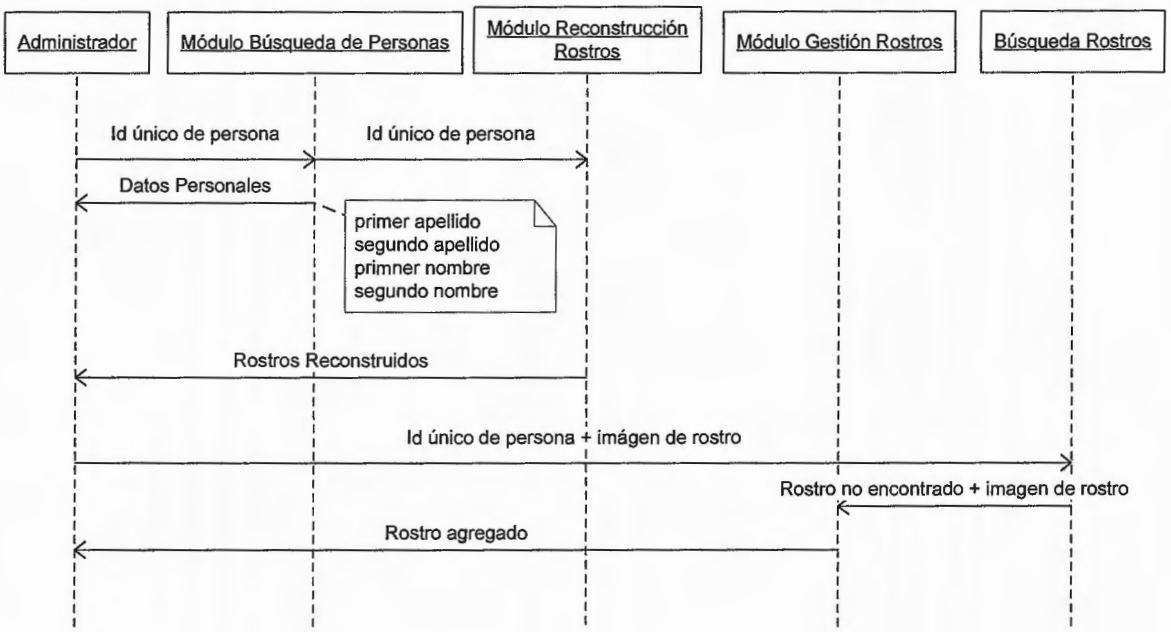
### *12.2.3.5 Escenario Eliminar Persona (Alternativo 1)*



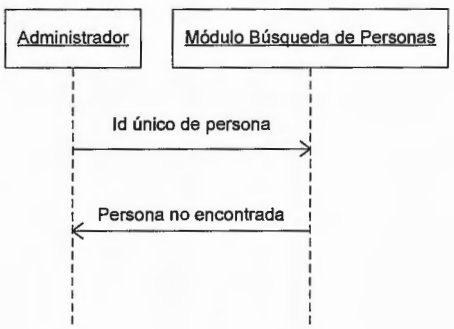

## 12.2.4. Diagrama de Secuencia: Gestión de Rostros



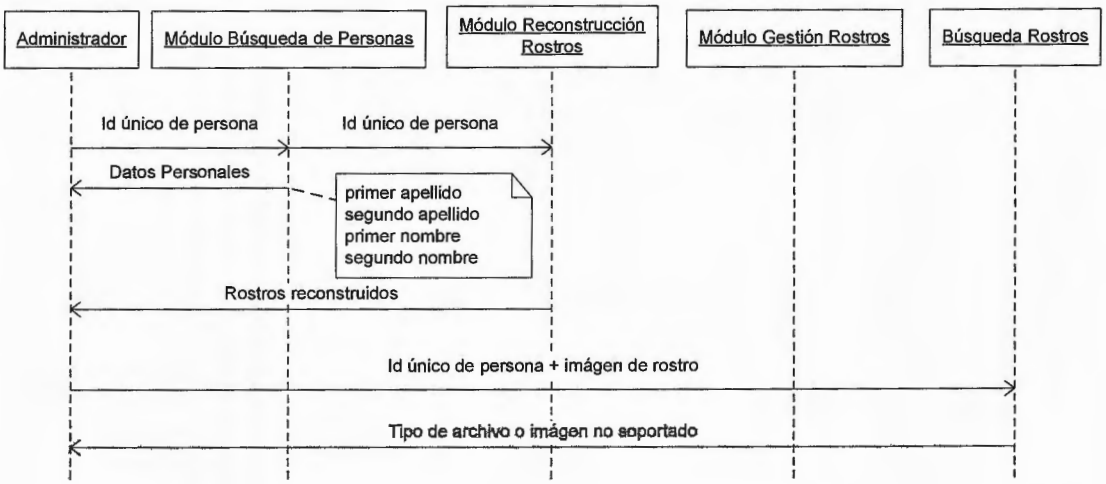
### 12.2.4.1 Escenario Agregar Rostro (Principal)



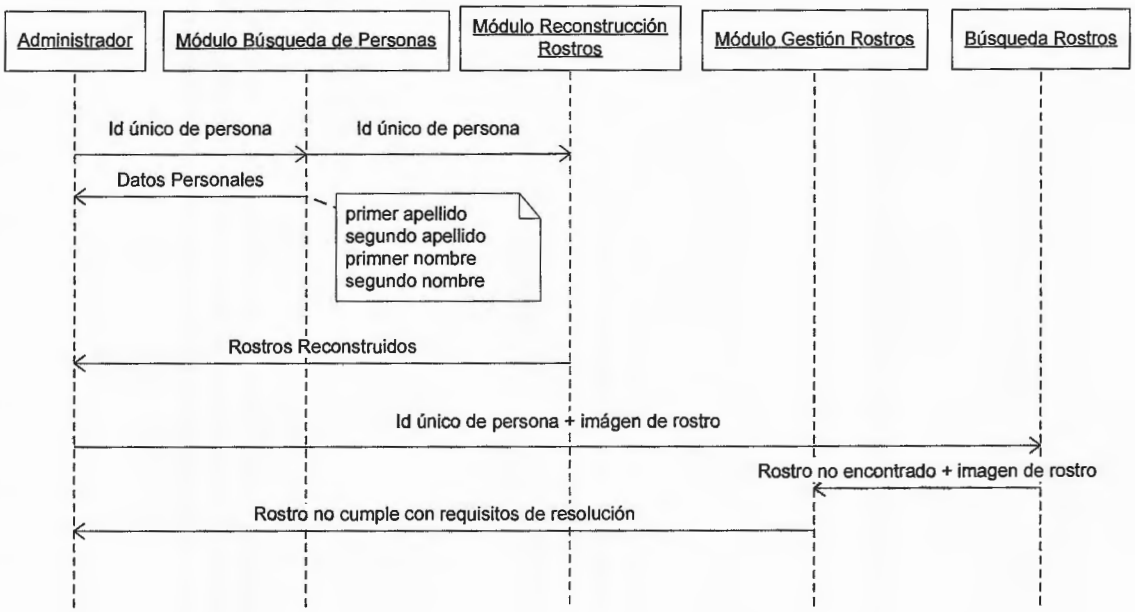

### 12.2.4.2 Escenario Agregar Rostro (Alternativo 1):



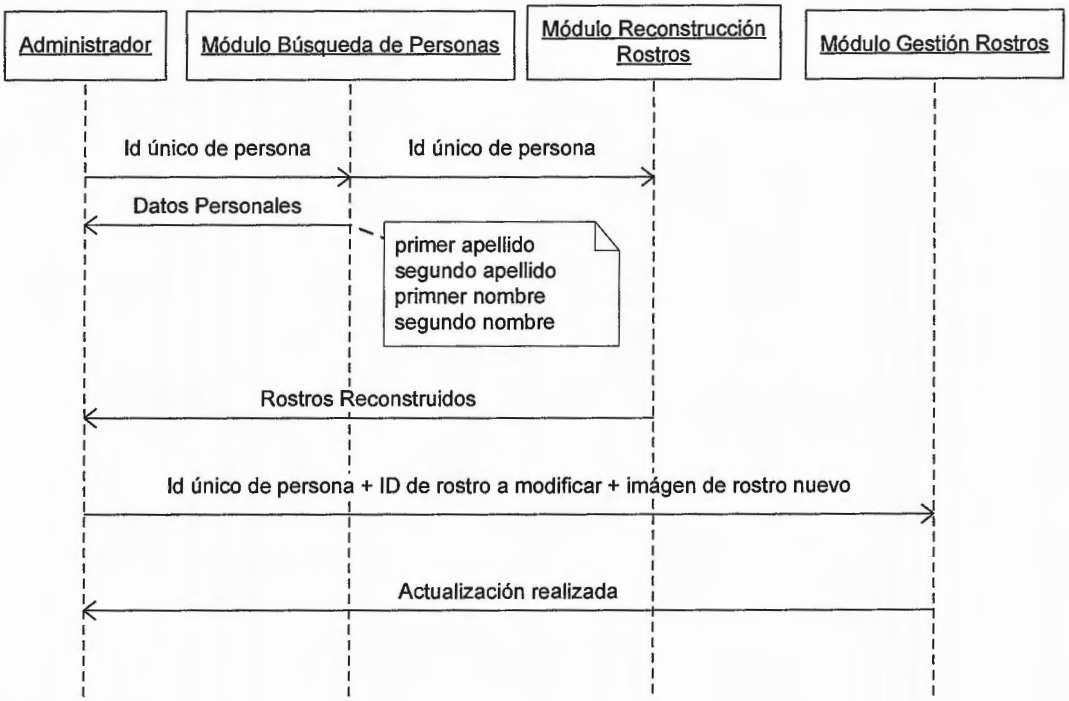
## *12.2.4.3 Escenario Agregar Rostro (Alternativo 2):*



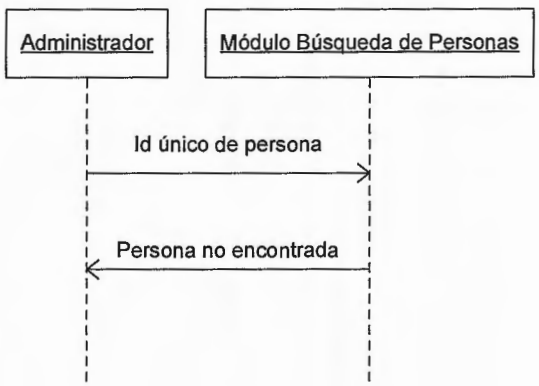

## *12.2.4.4. Escenario Agregar Rostro (Alternativo 3)*



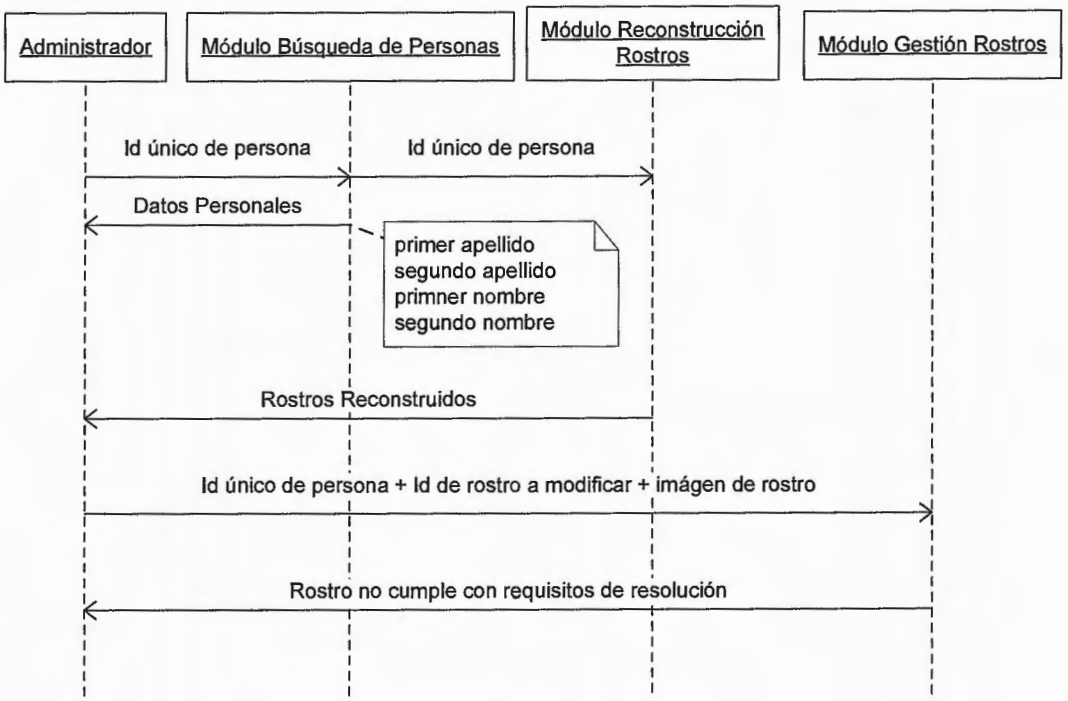
## 12.2.4.5. Escenario Actualizar Rostro (Principal)



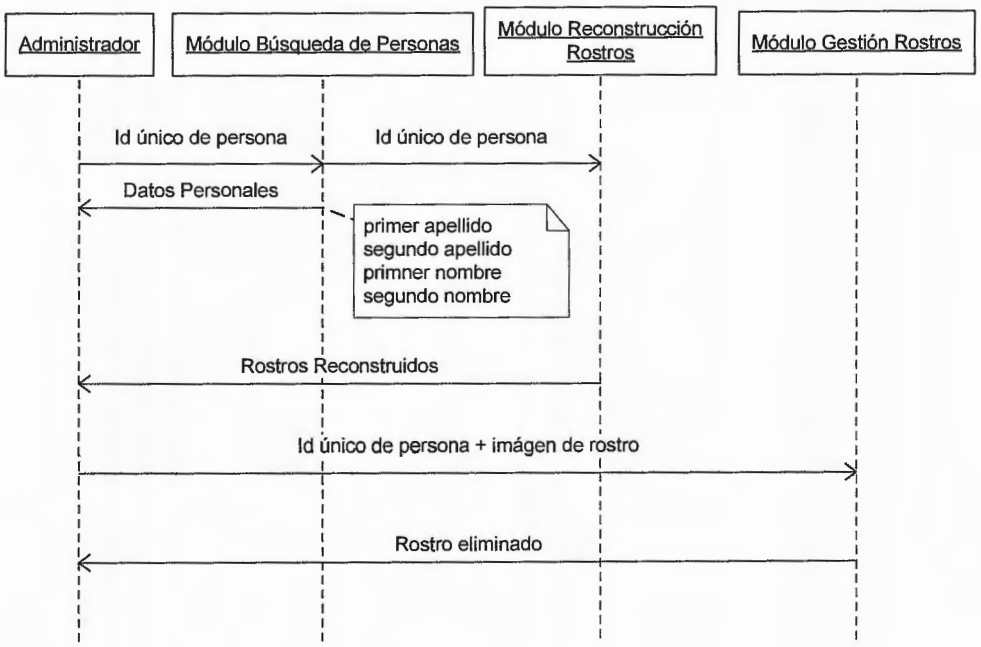

## 12.2.4.6. Escenario Actualizar Rostro (Alternativo 1)



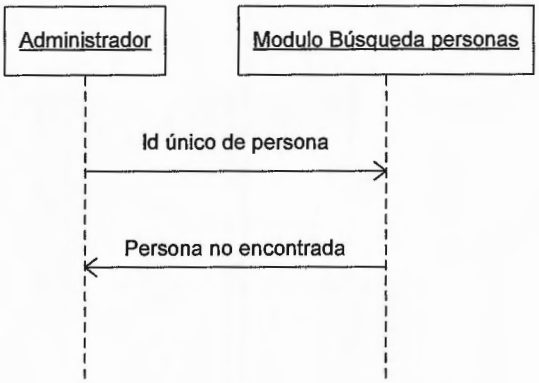
## 12.2.4.7. Escenario Actualizar Rostro (Alternativo 2)



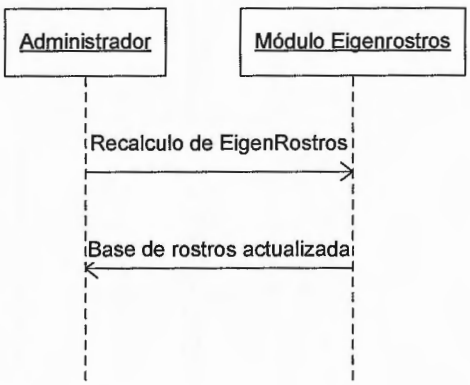

## 12.2.4.8. Escenario Eliminar Rostro (principal)



### *12.2.4.9. Escenario Eliminar Rostro (alternativo)*




## 12.2.5. Diagrama de Secuencia: Cálculo de Eigenrostros



## *12.3. Diseño del modulo PCA*

El módulo PCA es el encargado de encapsular las operaciones matemáticas requeridas; dicho módulo será implementado a partir de la técnica introducida por M. Turk y A. Pendtland; dicha técnica es necesaria para evitar la complejidad computacional que implica las operaciones con matrices de dimensiones extensas.

El módulo implementa la siguiente funcionalidad:

1. Cálculo de la imagen promedio: $\Psi = \frac{1}{M}\sum_{j=0}^{M-1}\Gamma_j$
2. Cálculo de la imagen original – la imagen promedio: $\Phi_i = \Gamma_i - \Psi$
3. Cálculo de la matriz A de NxM dimensiones según M. Turk y A. Pendtland (contiene vectores imágenes de promedio nulo) : $A = [\Phi_0\ \Phi_1\ \Phi_2\ \Phi_3 \ldots \Phi_{M-3}\ \Phi_{M-2}\ \Phi_{M-1}]$
4. Cálculo de la matriz de covarianza: $C = \frac{1}{M}\sum_{i=0}^{M-1}\Phi_i\Phi_i^T = \frac{1}{M}AA^T$
5. Cálculo de los eigenvectores y eigenvalores de la matriz de covarianza: $A^T A v_i = \mu_i v_i$.
6. Cálculo de los eigenvectores y eigenvalores a partir de los eigenvectores y eigenvalores encontrados de la matriz reducida: $u_i = \frac{1}{\sqrt{\mu_i}}Av_i$ y $\lambda_i = \frac{1}{M}\mu_i$
7. Cálculo de las proyecciones de una imagen desconocida sobre los eigenvalores.
8. Cálculo de las distancias euclideas entre las proyecciones de una imagen desconocida y las proyecciones originales.

Internamente, el módulo PCA contiene dos clases las cuales son:

| Módulo: | **PCA** |
|---|---|
| Clase: | **VectorPromedio** |
| Atributo/Método | Descripción |
| def __init__(self) | Constructor público de clase<br>Retorna: ningún valor<br>Parámetros:<br>self: parámetro de la clase (implícito) |
| def Promediar(self,vector) | Método público para promediar vectores<br>Retorna: ningún valor<br>Parámetros:<br>self: parámetro de la clase (implícito)<br>vector: imagen sectorizada que se desea promediar (numpy.array) |
| def Promedio(self) | Método público: retorna vector promedio.<br>Retorna: vector promedio (numpy.array)<br>Parámetros:<br>self: parámetro de clase (implícito) |
| Def Diferencia(self,vector): | Método público: calcula vectores imagen con media cero.<br>Retorna: Diferencia entre vector original y vector promedio.<br>Parámetros:<br>self: parámetro de clase (implícito) |

| Módulo: | **PCA** |
|---|---|
| Clase: | **VectorPromedio** |
| Atributo/Método | Descripción |
| | vector: imagen vectorizada original (numpy.array) |

| Módulo: | **PCA** |
|---|---|
| Clase: | **KLT** |
| Atributo/Método | Descripción |
| def __init__(self) | Constructor público de clase<br>Retorna: ningún valor<br>Parámetros:<br>self: parámetro de clase (implícito) |
| def ConcatenarVector(self,vector): | Método público: forma matriz de media nula.<br>Retorna: ningún valor<br>Parámetros:<br>self: parámetro de clase (implícito)<br>vector: vector de media nula (vector original - vector promedio) |
| def CalcularCovarianza(self) | Método público: calcula la covarianza de la matriz de vectores con media nula.<br>Retorna: ningún valor<br>Parámetros:<br>self: parámetro de clase (implícito) |

| Módulo: | **PCA** |
|---|---|
| Clase: | **KLT** |
| Atributo/Método | Descripción |
| def CalcularEigenRed(self) | Método público: calcula eigenvectores y eigenvalores de la matriz de covarianza reducida (M. Turk y A. Pendtland)<br>Retorna: ningún valor<br>Parámetros:<br>self: parámetro de clase (implícito) |
| def CalcularEigenVectores(self) | Método público: calcula la matriz de eigenvectores a partir de los eigenvectores de la matriz de covarianza reducida (M. Turk y A. Pendtland)<br>Retorna: ningún valor<br>Parámetros:<br>self: parámetro de clase (implícito) |
| def CalcularEigenvalores(self) | Método público: calcula la matriz de eigenvalores a partir de la matriz de eigenvalores de la matriz de covarianza reducida (M. Turk y A. Pendtland)<br>Retorna: ningún valor<br>Parámetros:<br>self: parámetro de clase |

| Módulo: | **PCA** |
|---|---|
| Clase: | **KLT** |
| Atributo/Método | Descripción |
| | (implícito) |
| def CalcularEigenvector(self,i) | Método público: calcula el i-ésimo eigenvector a partir del i-ésimo eigenvector de la matriz de covarianza reducida (M. Turk y A. Pendtland).<br>Retorna: eigenvector<br>Parámetros:<br>self: parámetro de clase (implícito)<br>i: índice del eigenvector que se desea calcular |
| def Eigenface(self,i) | Método público: retorna la i-ésima eigenface calculada.<br>Retorna: vector eigenface<br>Parámetro:<br>self: parámetro de clase (implícito)<br>i: índice del i-ésimo eigenface que se desea obtener. |
| def SecuenciaEigenvaluesRepr(self,n) | Método público: retorna una tupla de los n eigenvalores más representativos (orden descedente)<br>Retorna: tupla de n |

| Módulo: | **PCA** |
|---|---|
| Clase: | **KLT** |
| Atributo/Método | Descripción |
| | posiciones de los eigenvalores representantivos<br>Parámetros:<br>self: parámetro de clase (implícito)<br>n: cantidad entera de eigenvalores mas representativos |
| def ReconstruirImagen(self,vp,j,n) | Método público: reconstruye la j-ésima imagen a partir del vector promedio y una cantidad n de eigenfaces más representativos<br>Retorna: vector imagen reconstruida a partir de la suma del vector imagen promedio más los n eigenfaces más representativos.<br>Parámetros:<br>self: parámetro de clase (implicito)<br>vp: vector de la imagen promedio.<br>j: j-ésimo imagen que se desea reconstruir.<br>n: cantidad de eigenfaces |

| Módulo: | **PCA** |
|---|---|
| Clase: | **KLT** |
| Atributo/Método | Descripción |
| | mas representativas que se emplearan para la reconstrucción de la imagen. |
| | |

Adicionalmente el módulo PCA contiene un atributo público denominado dim_org el cual es una tupla que especifica las dimensiones originales de las imágenes utilizadas para el entrenamiento; la modificación/lectura de dicho atributo se realiza de la siguiente forma:

```
import PCA
PCA.dim_org=(256,256)
```

El atributo público PCA.dim_org es utilizado por las funciones ArchivoImagenAVector y VectorAArchivoImagen, tal como se especifica a continuación:

| Módulo: | **PCA** |
|---|---|
| Función | Descripción |
| def ArchivoImagenAVector(archivo) | Función de módulo pública la cual lee un archivo gráfico y retorna un vector de la imagen. |
| def VectorAArchivoImagen(archivo,vector) | Función de módulo pública la cual |

| | |
|---|---|
| | transforma un vector imagen en un archivo gráfico (de dimensiones PCA.dim_org) |

## *12.5 Bases de imágenes utilizadas para desarrollo del prototipo.*

Para el desarrollo del prototipo se esta trabajando con la base de datos de rostro de la Universidad de Yale[14]; la cual es un conjunto de 15 personas con 11 imágenes por rostro; dicha base de datos se puede descargar gratuitamente de la siguiente dirección de Internet:

ftp://plucky.cs.yale.edu/%2f/CVC/pub/images/yalefaces/yalefaces.tar

Existen otras bases de datos, sin embargo es necesario cumplir con ciertos requisitos para descargarlas, los sitios de donde se pueden encontrar mayor información al respecto son los siguientes:

Base de datos AR: http://cobweb.ecn.purdue.edu/RVL/database.htm

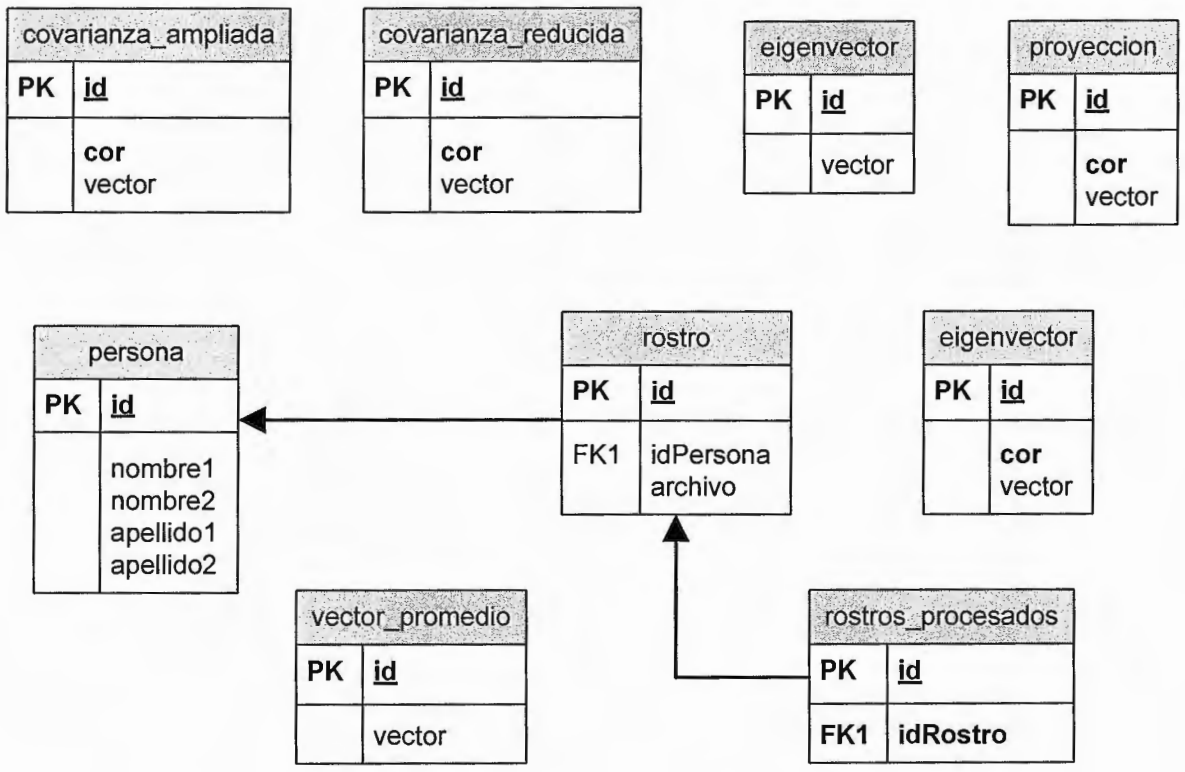
Base de datos FERET:
http://www.itl.nist.gov/iad/humanid/feret/feret_master.html

[14] Universidad de Yale, EE.UU.: www.yale.edu

## *12.6. Diagrama de la Base de Datos*

El prototipo esta diseñado partiendo del siguiente diagrama Entidad Relación; las tablas covarianza_ampliada, covarianza_reducida, eigenvector, proyeccion, eigenvector, vector_promedio constituyen tablas de almacenamiento de datos, los cuales no deben ser recalculados cada vez que inicia el programa (serialización de las matrices respectivamente).

Una persona puede tener varios rostros asociados, esto es muy recomendable para lograr un sistema de detección robusto que pueda adaptarse fácilmente a imágenes en diferentes ángulos y con diferentes accesorios (de una forma similar al reconocimiento humano de rostros).



La base de datos que se empleará sera MySQL; las tablas son de tipo ISAM (especializadas para el almacenamiento y recuperación rápida de la información; ya que probablemente se almacenen 500,000 o más elementos de matrices).

La conexión del sistema hacia la base de datos, se realizará a través de la librería genérica: ADOdb[15] la cual es una abstracción al acceso a base de datos basándose en el modelo implementado por Microsoft en sus lenguajes comerciales.

Actualmente ADOdb soporta las siguientes bases de datos:

| Class | Requirimientos | Notas |
|---|---|---|
| odbc | Download PythonWin extensio | No support for SelectLimit, UpdateBlob, UpdateBlobFile, Insert_ID, RecordCount and Affected_Rows. |
| access | Requires mxodbc. | Only SelectLimit( ) with no *offset* parameter supported. RecordCount( ) not supported. |
| mssql | Requires mxodbc. | Only SelectLimit( ) with no *offset* parameter supported. RecordCount( ) not supported. |
| mysql | Download MySQL-python extension | |
| mxodbc | Superior odbc extension. Licensing fee required for commercial us | SelectLimit( ) not supported. RecordCount( ) not supported. |
| mxoracle | Requires mxodbc. Connect to Oracle using ODBC. | Only SelectLimit( ) with no *offset* parameter supported. Requires Oracle client installed. RecordCount( ) not supported. |
| oci8 | Download cx_Oracle extension. Also | Despite the name, it works with Oracle |

[15] ADOdb: http://adodb.sourceforge.net/

| | | |
|---|---|---|
| | requires Oracle client to be installed. | 8, 9 and later. SelectLimit( ) does not support the *offset* parameter. RecordCount( ) not supported. |
| odbc | Download PythonWin extension | SelectLimit( ) only works with Access,VFP and Microsoft SQL Server. The *offset* parameter is not supported. No support for UpdateBlob, UpdateBlobFile, Insert_ID, RecordCount and Affected_Rows. |
| odbc_mssql | Download PythonWin extension | Same limitations as adodb_odbc extension, except that Insert_ID and Affected_Rows supported. |
| postgres | Download psycopg extension | |
| vfp | Requires mxodbc. | |
| sqlite | Requires pysqlite. | Contributed by Glenn Washburn. |

## *12.7. Script de Creación de la Base de Datos*

A continuación se anexa script de creación de base de datos PCA, requerida para el almacenamiento de rostros y eigenvectores.

```
-- MySQL Administrator dump 1.4
--
-- ------------------------------------------------------
-- Server version 5.0.24a-community-nt


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;

/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO'
*/;


--
-- Create schema pca
```

```
--

CREATE DATABASE IF NOT EXISTS pca;
USE pca;

--
-- Definition of table `covarianza_ampliada`
--

DROP TABLE IF EXISTS `covarianza_ampliada`;
CREATE TABLE `covarianza_ampliada` (
  `id` int(10) unsigned NOT NULL auto_increment,
  `cor` int(10) unsigned NOT NULL,
  `vector` longblob,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;



--
-- Definition of table `covarianza_reducida`
--

DROP TABLE IF EXISTS `covarianza_reducida`;
CREATE TABLE `covarianza_reducida` (
  `id` int(10) unsigned NOT NULL auto_increment,
  `cor` int(10) unsigned NOT NULL,
  `vector` longblob,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=37 DEFAULT CHARSET=latin1;


--
-- Definition of table `eigenvalor`
--

DROP TABLE IF EXISTS `eigenvalor`;
CREATE TABLE `eigenvalor` (
  `id` int(10) unsigned NOT NULL default '0',
  `vector` longblob,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;


--
-- Definition of table `eigenvector`
--

DROP TABLE IF EXISTS `eigenvector`;
CREATE TABLE `eigenvector` (
  `id` int(10) unsigned NOT NULL auto_increment,
  `cor` int(10) unsigned default '0',
  `vector` longblob,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

```
--
-- Definition of table `persona`
--

DROP TABLE IF EXISTS `persona`;
CREATE TABLE `persona` (
  `id` int(10) unsigned NOT NULL auto_increment,
  `nombre1` varchar(45) NOT NULL,
  `nombre2` varchar(45) NOT NULL,
  `apellido1` varchar(45) NOT NULL,
  `apellido2` varchar(45) NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=23 DEFAULT CHARSET=latin1;


--
-- Definition of table `proyeccion`
--

DROP TABLE IF EXISTS `proyeccion`;
CREATE TABLE `proyeccion` (
  `id` int(10) unsigned NOT NULL auto_increment,
  `cor` int(10) unsigned NOT NULL,
  `vector` longblob,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=37 DEFAULT CHARSET=latin1;


--
-- Definition of table `rostro`
--

DROP TABLE IF EXISTS `rostro`;
CREATE TABLE `rostro` (
  `id` int(10) unsigned NOT NULL auto_increment,
  `idPersona` int(10) unsigned NOT NULL,
  `archivo` varchar(100) default NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=58 DEFAULT CHARSET=latin1;



--
-- Definition of table `rostros_procesados`
--

DROP TABLE IF EXISTS `rostros_procesados`;
CREATE TABLE `rostros_procesados` (
  `id` int(10) unsigned NOT NULL,
  `idRostro` int(10) unsigned NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;



--
```

```
-- Definition of table `vector_promedio`
--

DROP TABLE IF EXISTS `vector_promedio`;
CREATE TABLE `vector_promedio` (
  `id` int(10) unsigned NOT NULL default '0',
  `vector` longblob,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;



/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
```

# 13. MANUAL DE INSTALACION

La instalación de la aplicación puede realizarse en ambiente Windows ® o ambiente *NIX; el software necesario para el funcionamiento de la aplicación es el siguiente:

- Python 2.4 o superior
- WxPython 2.6 o superior
- Numpy for Python
- SciPy for Python
- MySQL extensions for Python.
- Python Adodb-201
- MySQL 5.0 o superior

Para realizar la instalación de cada uno de los componentes es desable tener un rol de administrador en la maquina o el password del superuser en el ambiente *NIX.

## *13.1 Instalación de Python con las librerías adicionales WxPython, Numpy, SCIPy.*

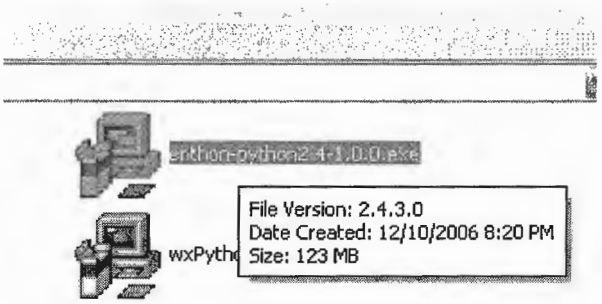
En el ambiente Windows, la distribución de Python se puede descargar de los siguientes sitios:

- http://www.python.org
- http://www.activestate.com/products/activepython/
- http://code.enthought.com/enthon/

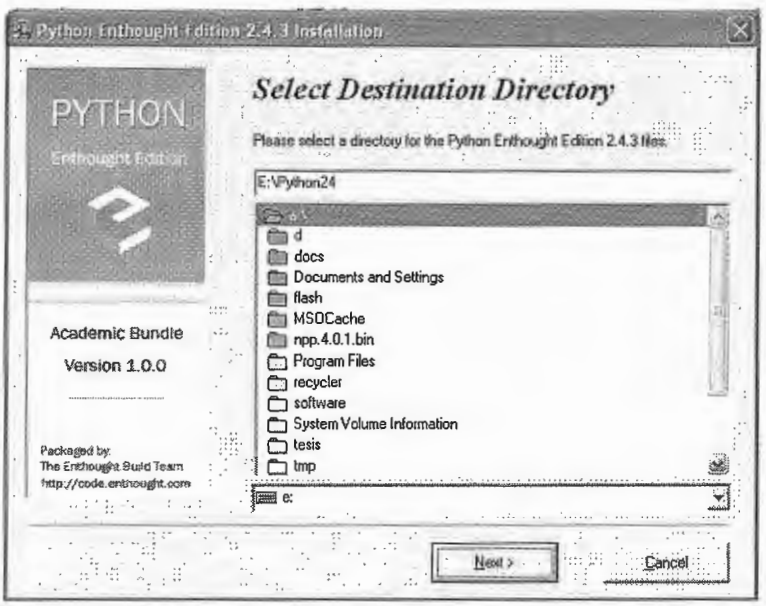
La diferencia entre una distribución de Python consiste en la cantidad y clase de librerías adicionales que incluye.

Se sugiere utilizar la versión descargable del sitio http://code.enthought.com/enthon ya que esta versión incluye Python, las librerías WxPython, Numpy y muchas otras orientadas hacia aplicaciones científicas; A continuación se presentan las pantallas de la instalación de esta distribución de Python:

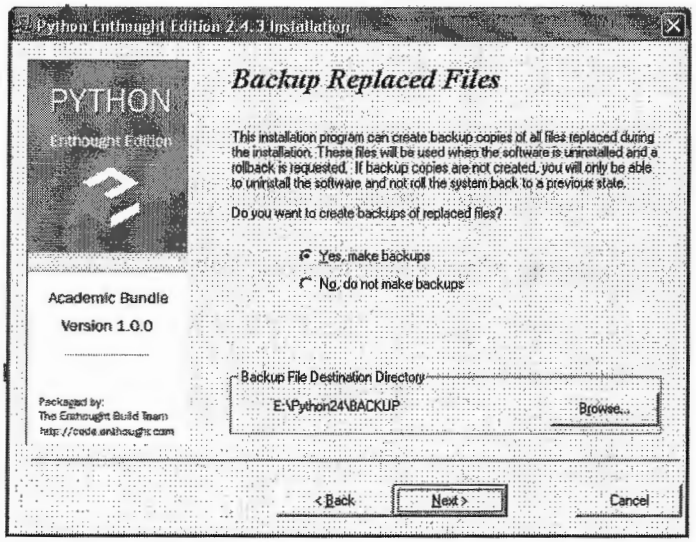
1. Ejecutar la aplicación del instalador de Python, el instalador de enthought python mide aproximadamente 123 MiB debido a las librerías adicionales que incluye:



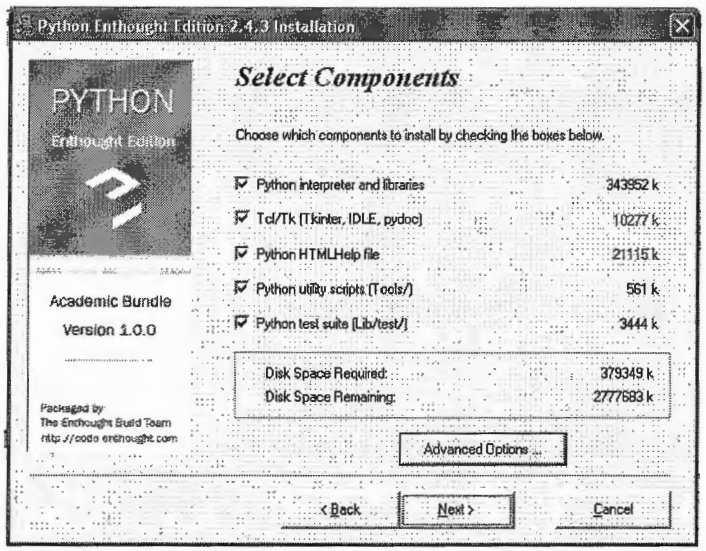

2. Selección de la ubicación a donde se instalará Python, normalmente se instala como un directorio independiente bajo la raíz del disco duro.



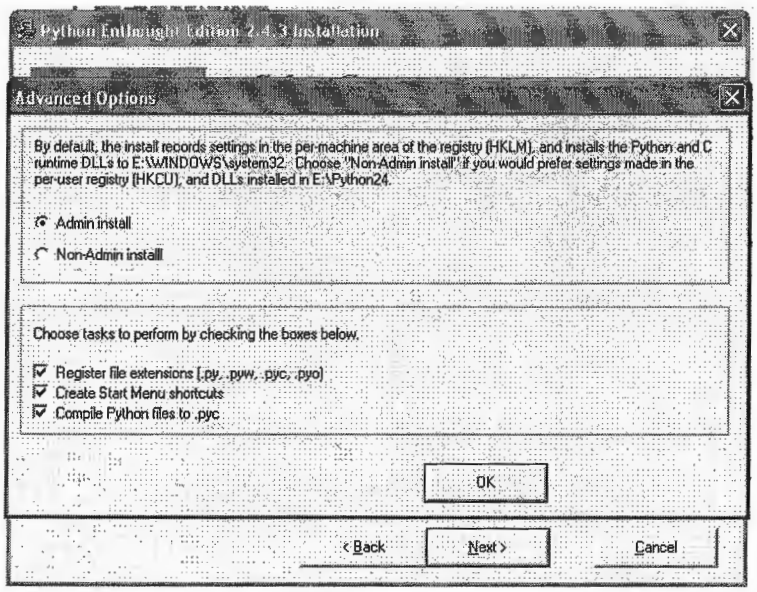
3. Como medida de seguridad, enthon python efectúa un backup de los archivos que pueden ser conflictivos o que reemplazará de una instalación anterior:



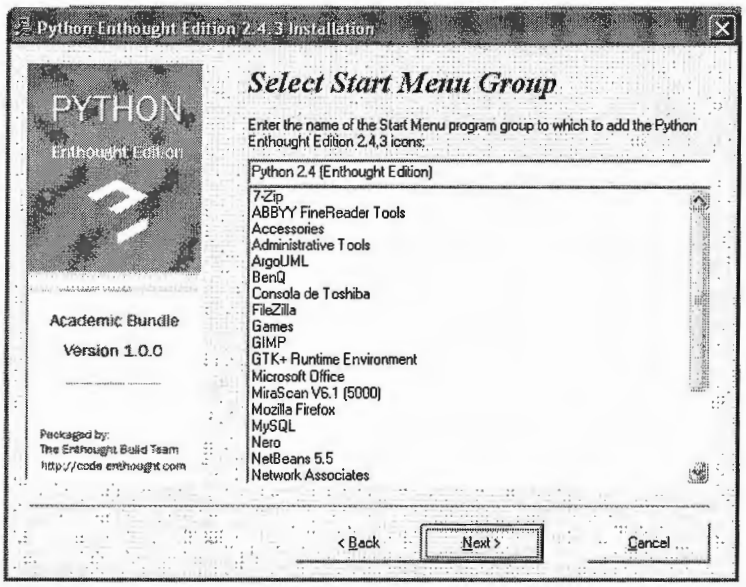

4. Enthonpython preguntará que componentes desea instalar de toda la distribución; se recomienda instalar todas las opciones lo cual no consumirá demasiado espacio en disco duro (aprox. 380 MiB)



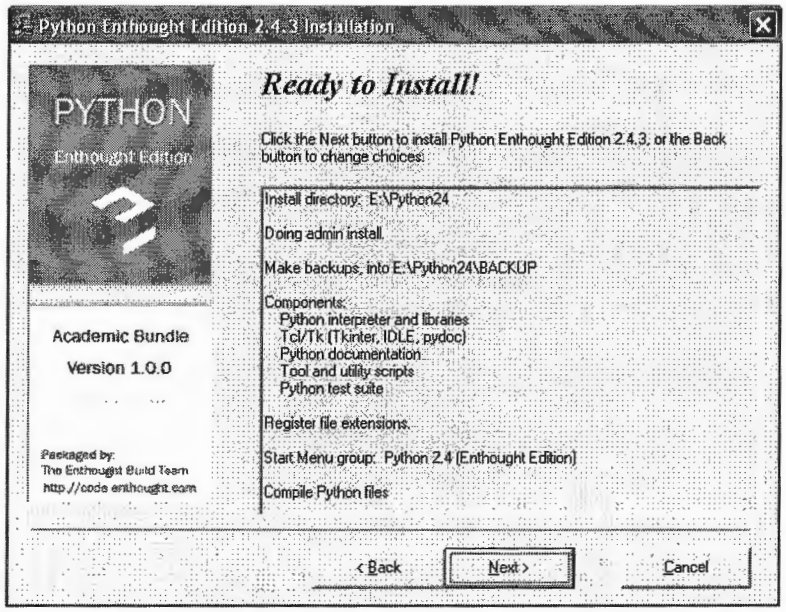
5. En las opciones avanzadas es preferible que sea una instalación administrativa (para que el interprete este disponible para todos los usuarios del equipo) y que se compilen los programas .py de las librerías en .pyc de esta forma se optimiza la ejecución de Python y el set de librerías incluídas.

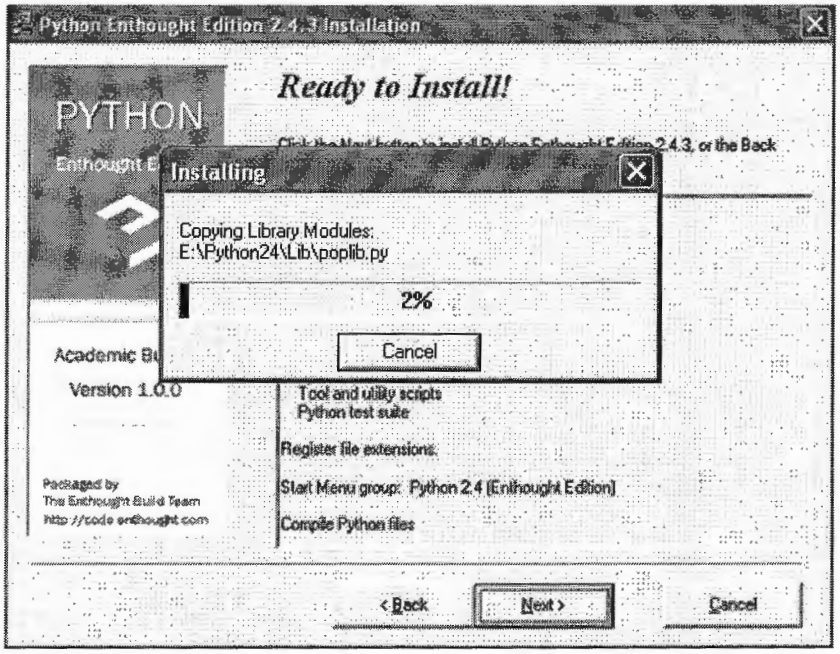



6. Se puede personalizar el grupo de programas que se creará en el menú de Windows o dejar el default:



7. Una vez completado los pasos anteriores, Python junto con un gran set de librerías cientificas y gráficas esta listo para instalarse en su equipo:

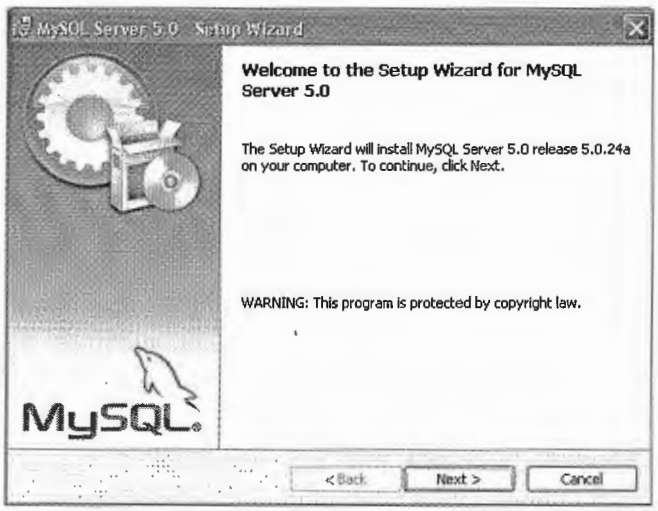


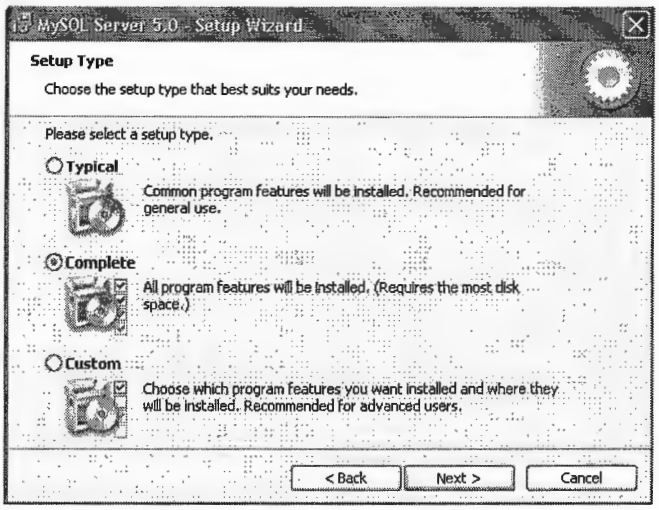
## *13.2. Instalación y configuración de MySQL 5.0*

Para la instalación del Servidor MySQL 5.0, este se puede descargar del sitio: http://dev.mysql.com/downloads/ donde se puede elegir la versión "Community Server". Los pasos de instalación del MySQL se ilustran a continuación:

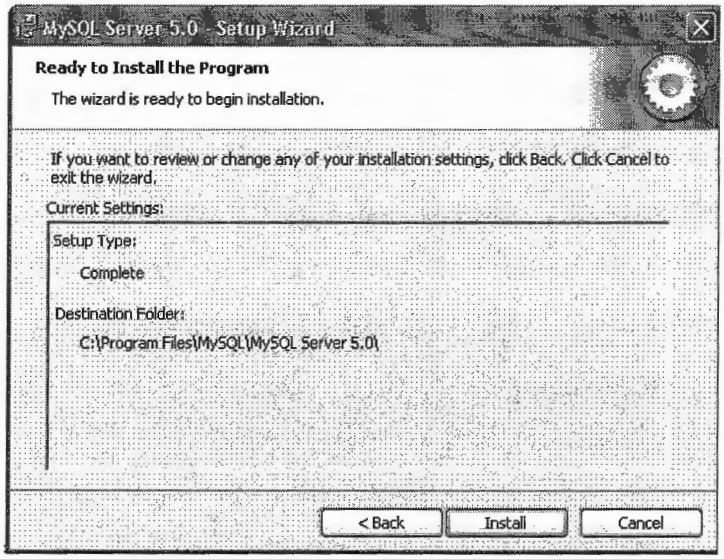
1. Inicio de Instalación:

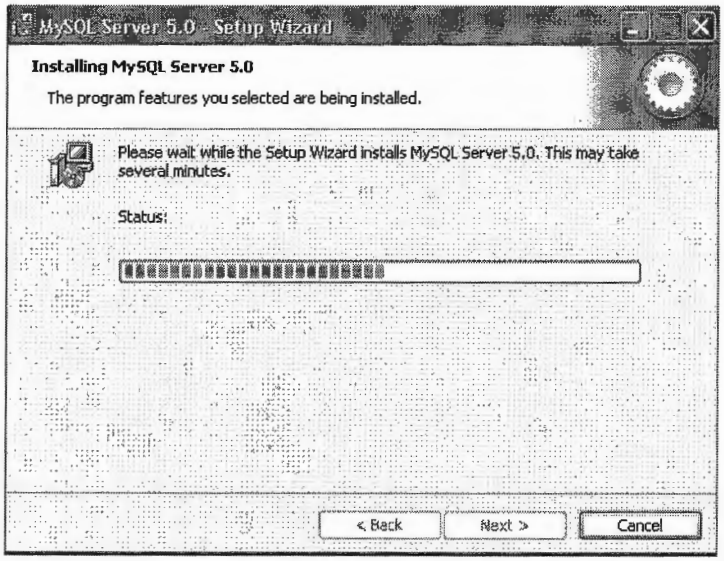



2. Selección de tipo de Instalación: Se recomienda instalar la versión completa.

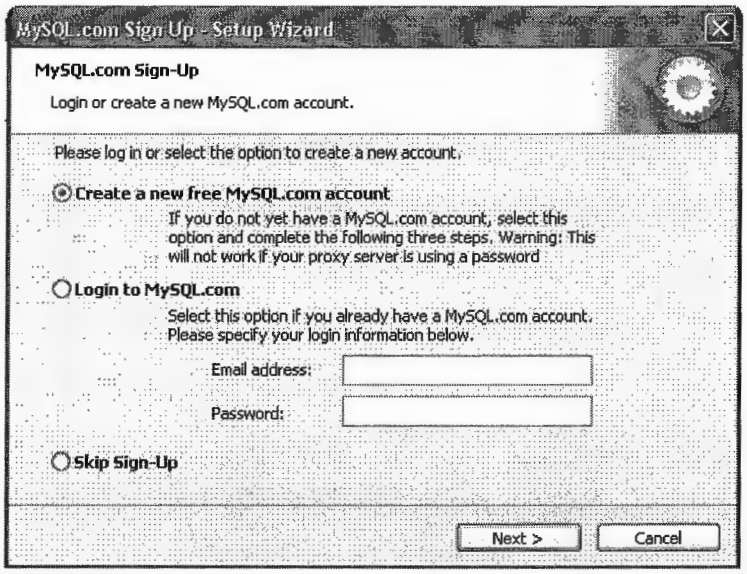


3. Pantalla final de instalación.

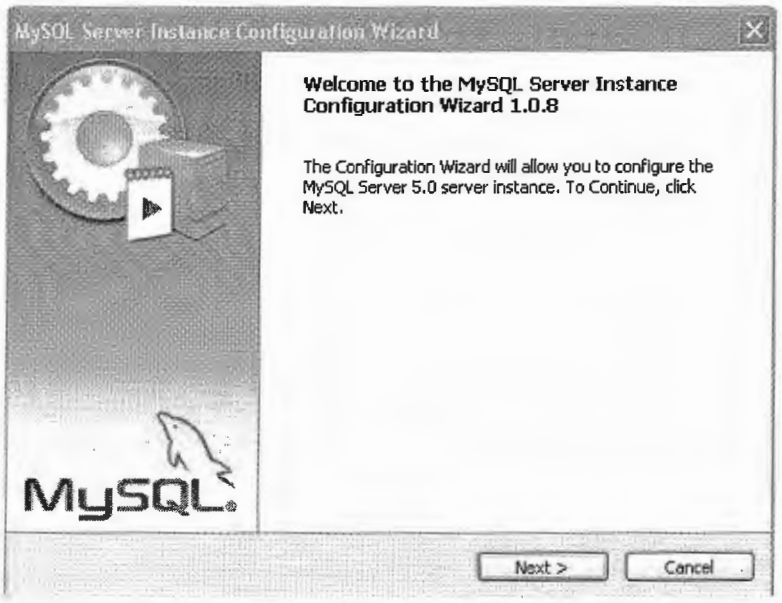


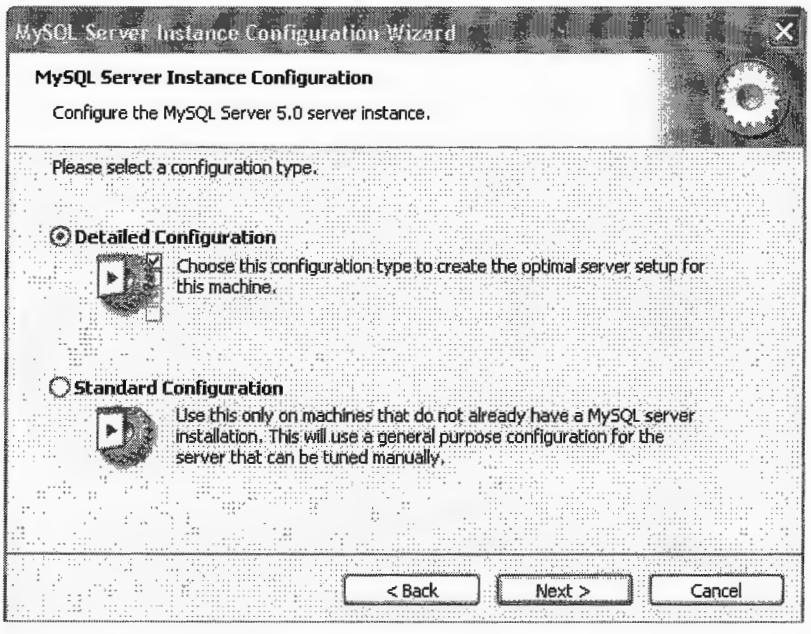

4. En la pantalla siguiente se puede crear una cuenta para el sitio MySQL.com donde se recibirán noticias sobre nuevas versiones; si no se dispone de Internet se debe seleccionar la opción: Skip Sign-Up.




5. El paso final en la Instalación de MySQL consiste en indicar que tipo de carga tendra el servidor, es decir la configuración del performance y utilización de memoria.



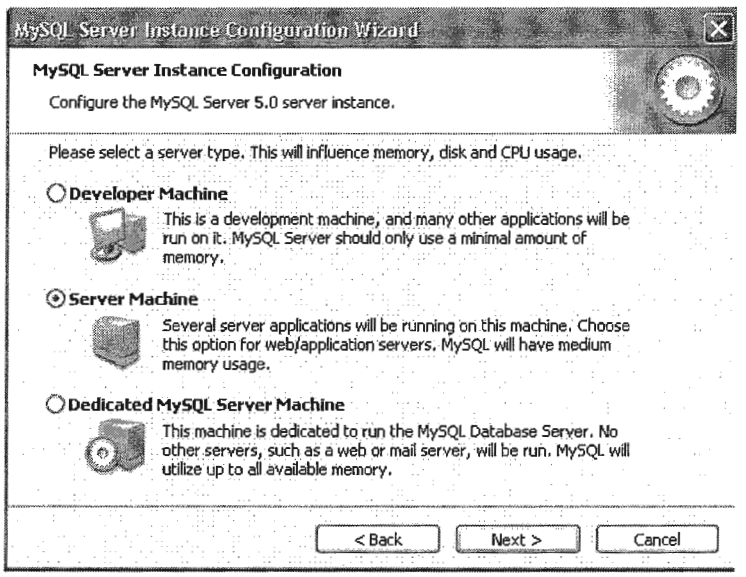

6. Se recomienda configurarlo detalladamente, asi se tiene control sobre las opciones del servidor.




7. El programa de instalación de MySQL permite realizar 3 tipos de configuración:

a. Developer Machine: Este tipo de configuración hace que el servidor ocupe poco recursos ya que no es un ambiente de producción.
b. Server Machine: Este tipo de configuración es para aplicaciones de mediana escala, tipicamente aplicaciones web o con pocos usuarios.
c. Dedicated MySQL Server Machine: En este caso, el Servidor SQL es para aplicaciones críticas o muy intensivas.

Para que el servidor no consuma demasiados recursos, se recomienda seleccionar la opción "Server Machine"

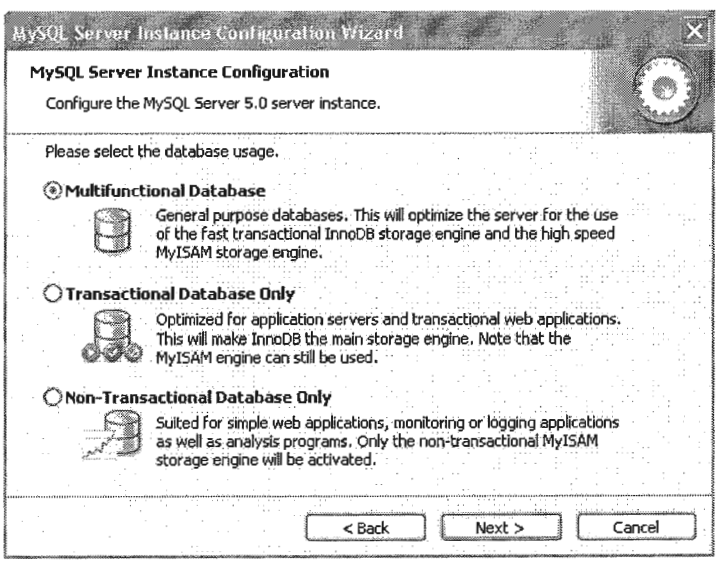



8. La siguiente pantalla se refiere a el tipo de bases de datos que soportará el Servidor MySQL; MySQL posee diferentes formatos para almacenar la información:
   a. el formato MyISAM es un formato diseñado especialmente para consultar la información lo cual le ha permitido a MySQL posicionarse como uno de los gestores de Bases de Datos más rápido en el mercado; sin

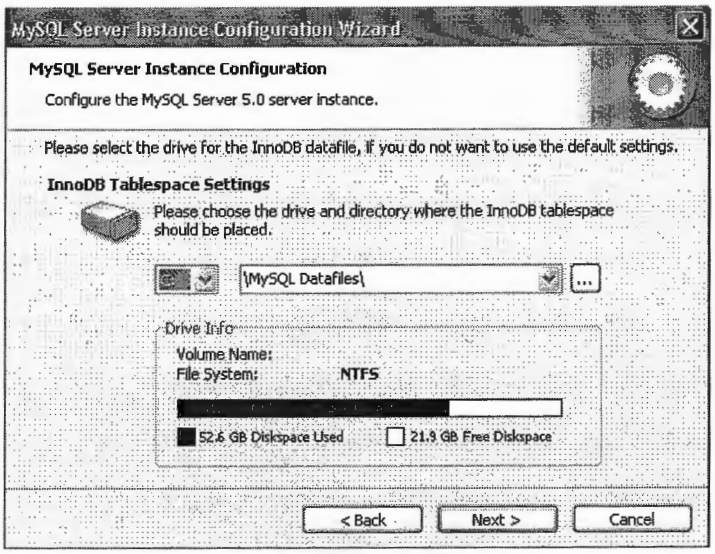
embargo este formato es pobre para establecer relaciones de integridad.

b. MySQL ademas del formato MyISAM, soporta InnoDB la cual esta diseñada para las aplicaciones que necesitan soporte a transacciones, Falcon (transaccional), Merge (recomendadas para tablas con muchos registros, Memory (tablas existentes únicamente en memoria), BDB (Berkeley DB), NDB (MySQL Clustering con soporte para transacciones y alta disponibilidad para aplicaciones críticas), Archive (tablas que son almacenadas de forma comprimida en el medio físico), CSV (formato delimitado por comas), Blackhole (tabla que no almacena ninguna información).

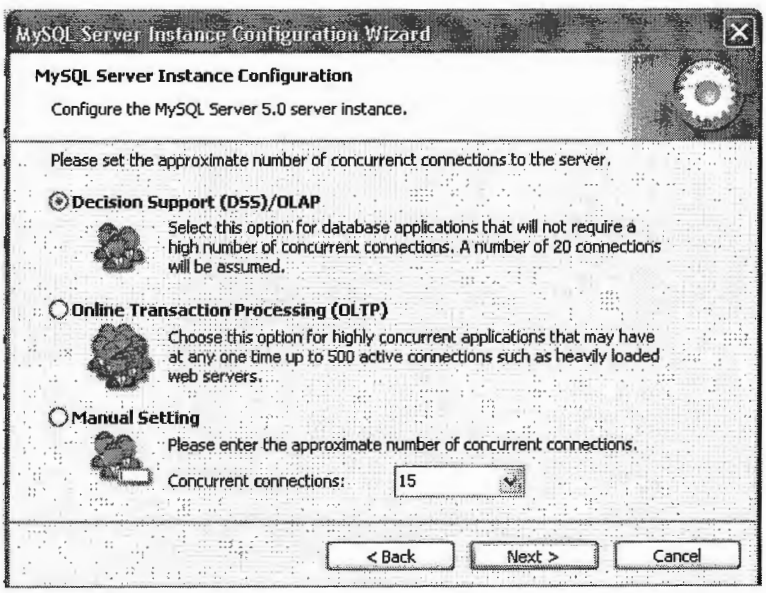
La opción que se recomienda es: "Multifunctional Database" de esta forma MySQL puede utilizar diferentes tipos de formatos para almacenar los datos según las necesidades; aunque la aplicación PCA utiliza exclusivamente las tablas MyISAM para obtener rendimiento en consultas.



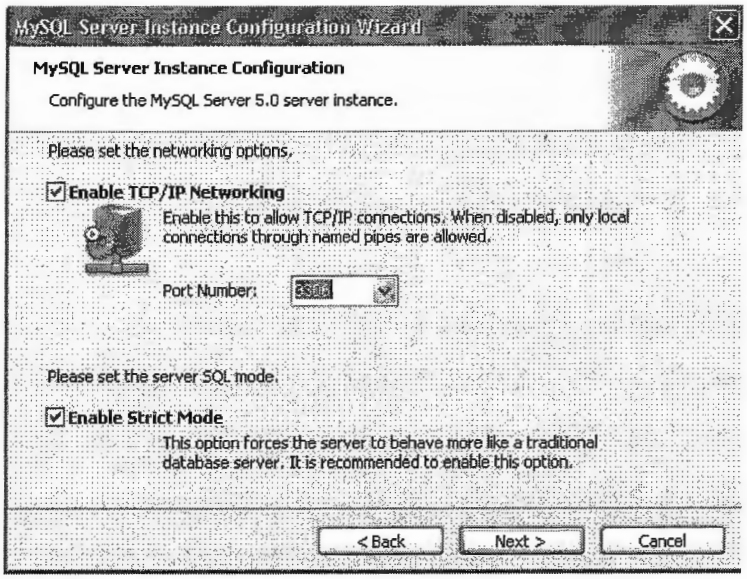
9. MySQL solicitará otro directorio donde almacenar los archivos en formato InnoDB



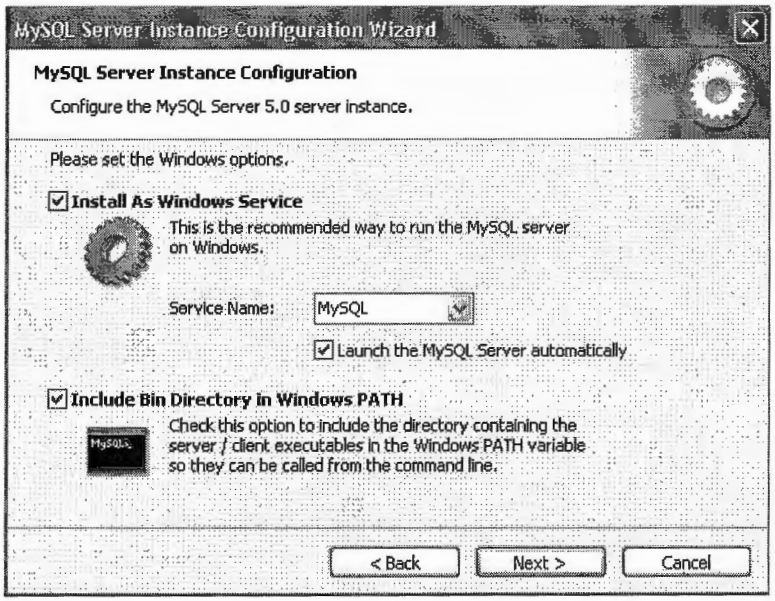

10. Posteriormente se debe indicar al MySQL un aproximado de las cargas de conexiones, indicar Decision Support (DSS) la cual implica no más de 20 conexiones simultáneas.




11. Es necesario especificar el puerto de conexión y habilitar el módo estricto,

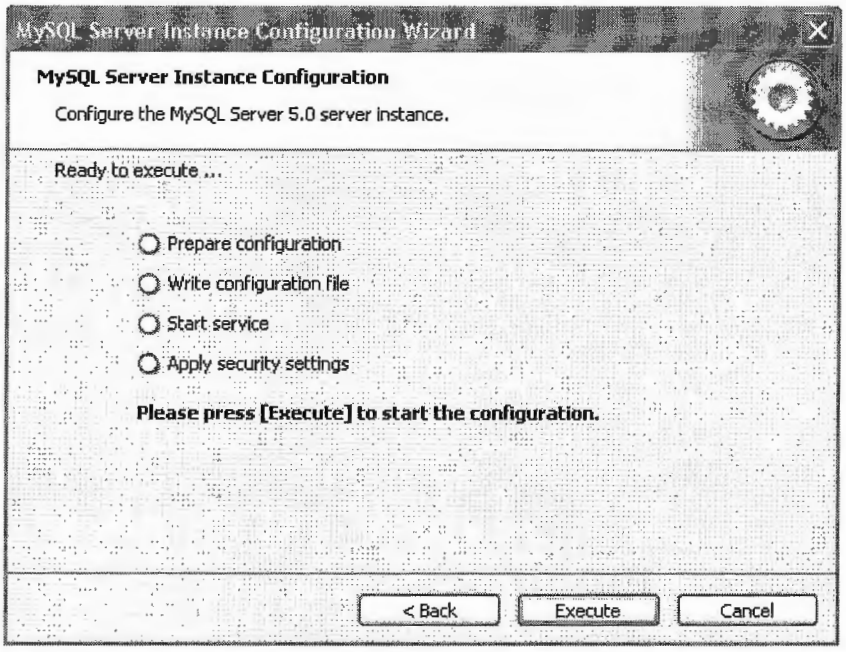



12. La etapa final de la configuración es decidir si MySQL lenvatará manualmente o como un servicio de Windows (se recomienda que levante como un servicio) e incluir la dirección de los ejecutables en la variable de entorno PATH.



13. Especificación del password del root o Secuty Administrator del servidor MySQL.


14. Finalización del Asistente de Configuración del MySQL.



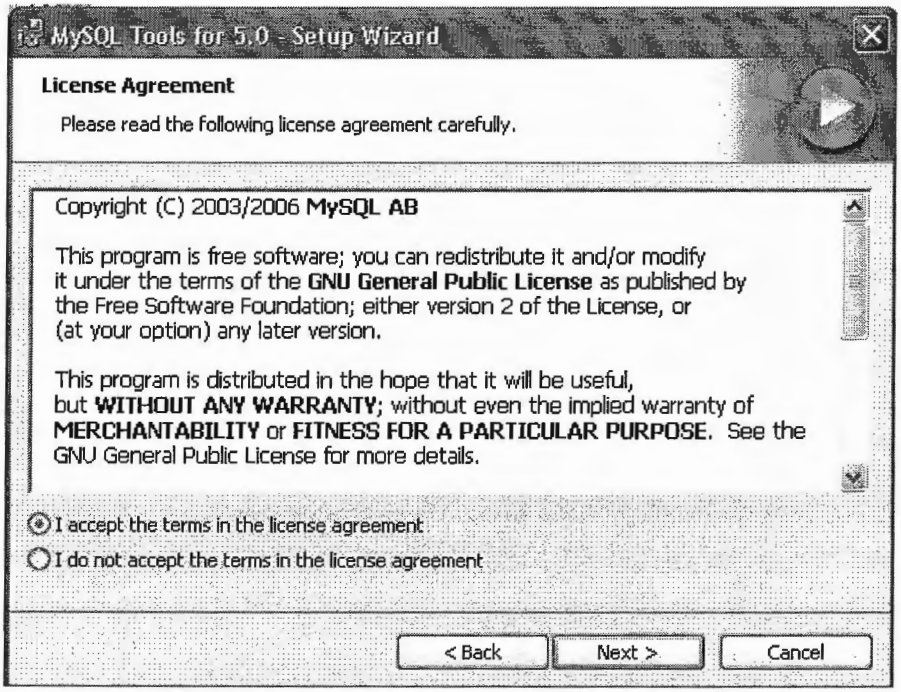
## *13.3. Instalación de las herramientas de Administración de MySQL*

Las herramientas de administración de MySQL permiten realizar todas las actividades relacionadas con el mantenimiento de un servidor; actividades entre las que destacan: administración de usuarios, administración de permisos y accesos sobre las bases de datos, operaciones de backup y restauración de bases de datos.

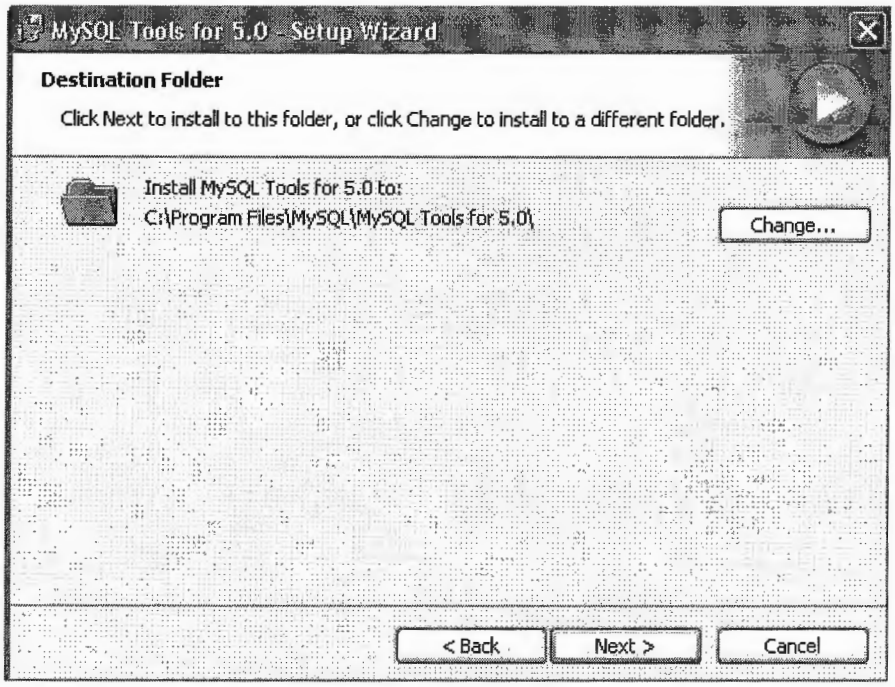
El sitio donde se pueden descargar las herramientas de administración es el siguiente: http://dev.mysql.com/downloads/gui-tools/5.0.html; los pasos de instalación se detallan a continuación:

1. Inicio de la Instalacion

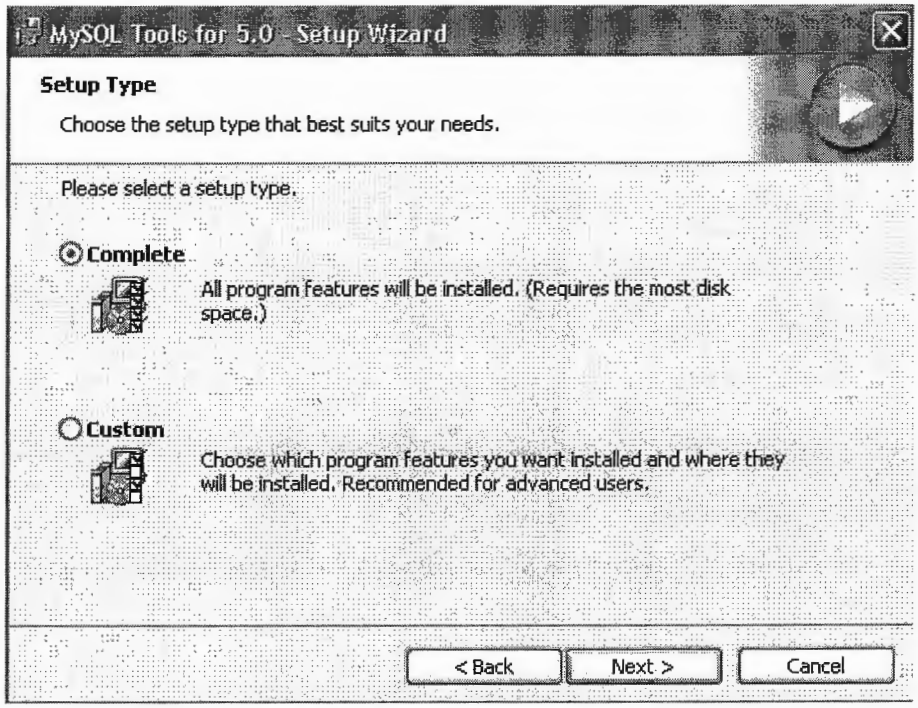
2. Seleccionar: Aceptar la Licencia.



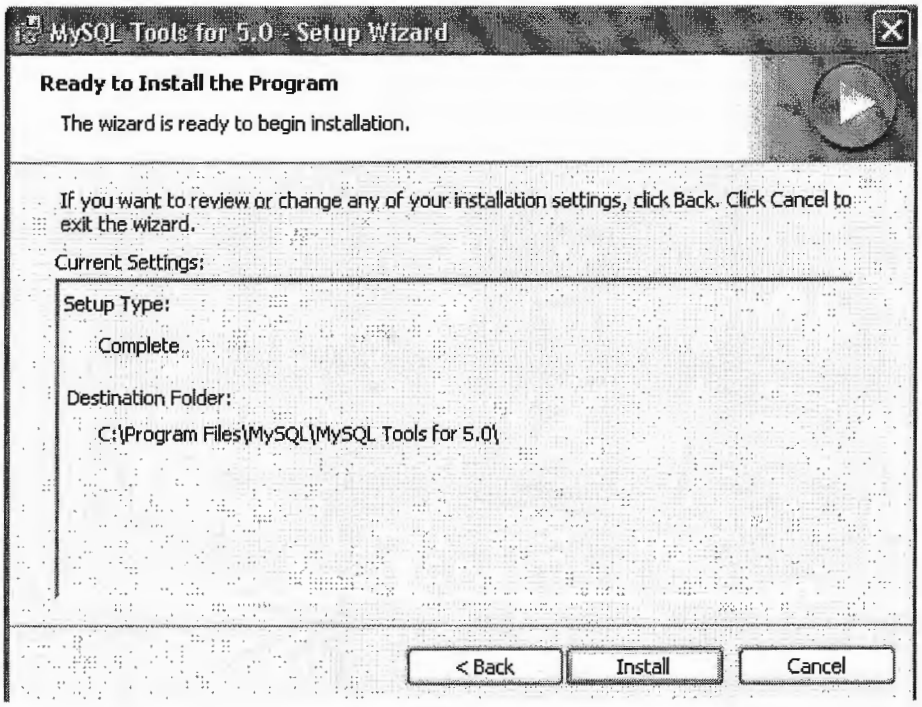

3. Selección del directorio donde se instalará el programa:



4. Selección del tipo de instalación.

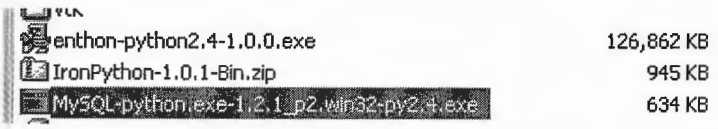



5. Pantalla final de la instalación.



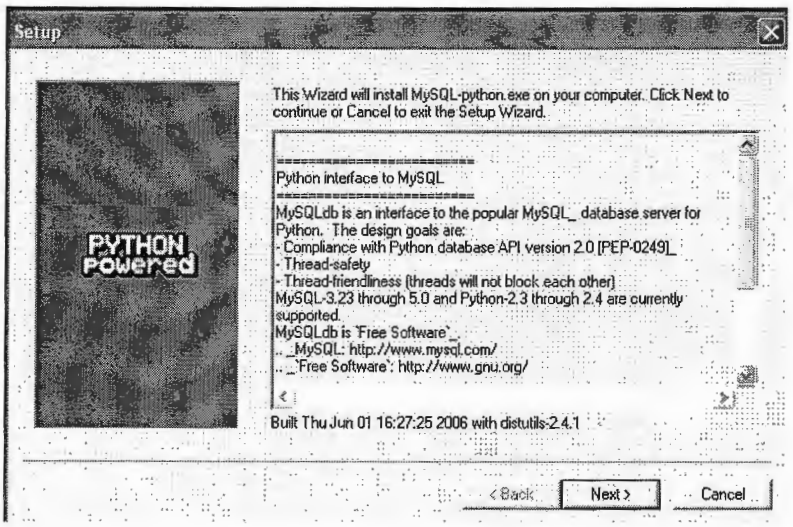
## *13.4. Instalación de MySQL-python*

La instalación de la Interfaz de conexión MySQL y Python se instala a través del software: MySQL-python.exe-1.2.1_p2.win32-py2.4.exe, el cual se descarga del sitio: www.mysql.com, mide aprox. 700 Kbi y requiere que se encuentre instalado en la máquina las librerías de conexión del MySQL.

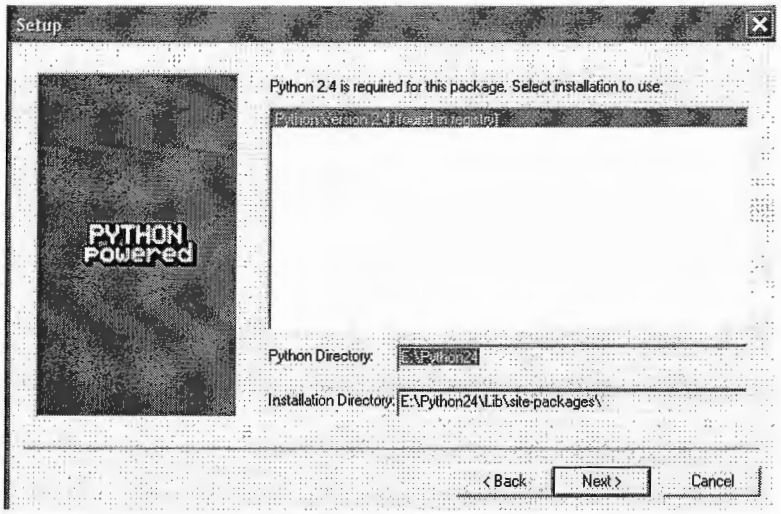



Los pasos para la instalación son los siguientes:

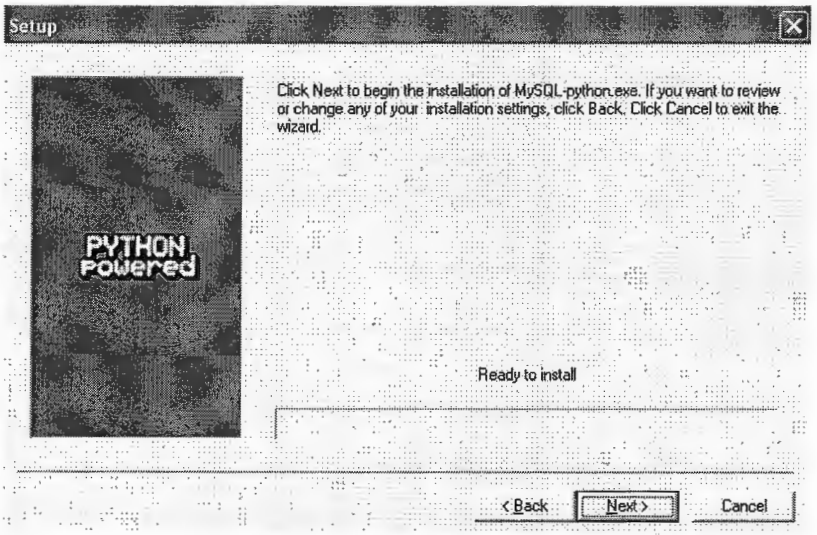
1. Inicio del instalador.




2. El instalador localizará la instalación de python existente e indicara si es esa versión la que se desea utilizar




3. La parte final de la instalación.



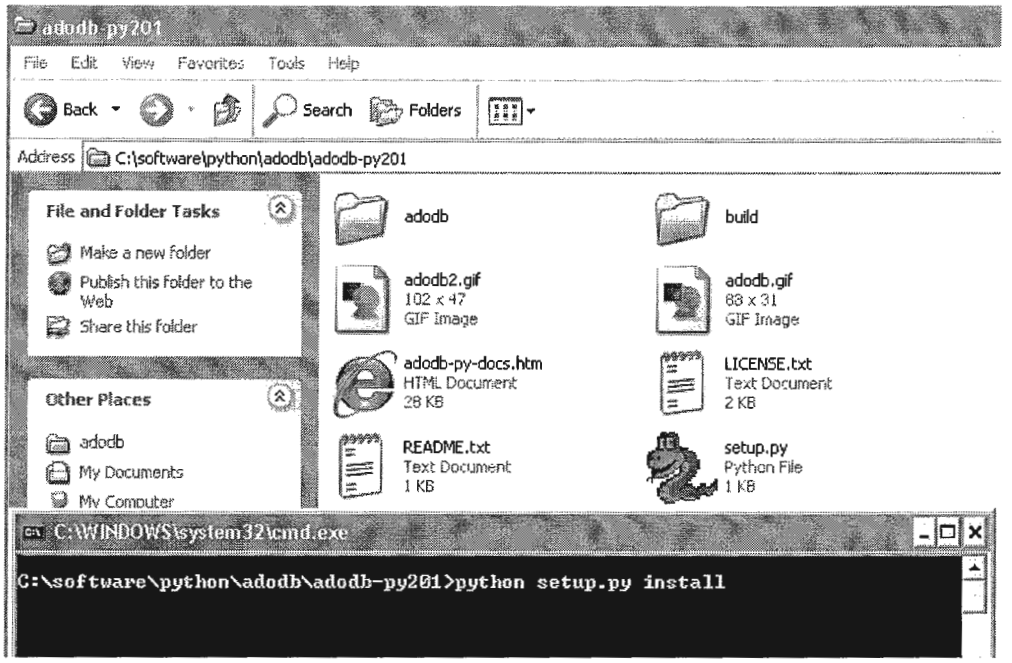

## *13.5. Instalación de ADOdb para Python*

ADOdb es una librería de abstracción para el acceso a bases de datos (modelada según las API de Microsoft), ADOdb fue desarrollado originalmente para PHP, luego fue portado a Python.

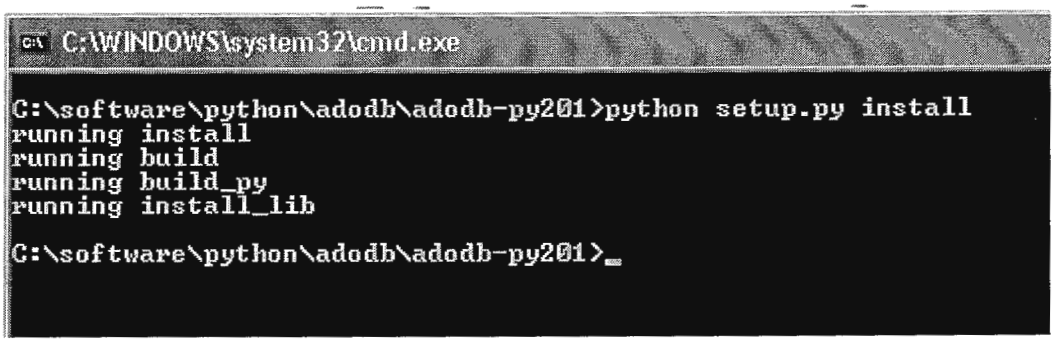
El licenciamiento de ADOdb es basado en la licencia BSD, y actualmente soporta las siguientes abstracciones:

- ODBC
- Access
- MS SQL
- MXodbc
- MXoracle
- OCi8
- ODBC_mssql
- Postgress
- Visual Fox Pro
- SQLite

La instalación no posee ambiente gráfico, por lo que es necesario descargar los archivos del sitio: http://phplens.com/lens/adodb/adodb-py-docs.htm, luego descomprimirlos bajo una carpeta y ejecutar las siguiente instrucción: python setup.py install (desde una ventana de comandos del ms-dos, o desde una terminal de Linux/Unix)



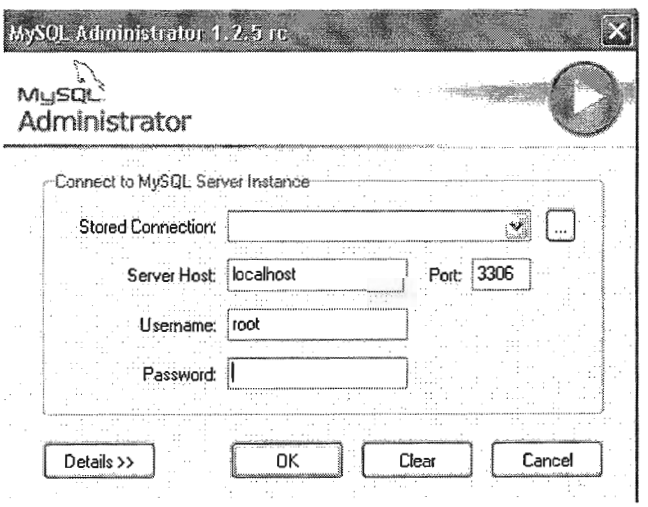
La instalación copiara los archivos adecuados bajo la estructura C:\Python24\Lib\site-packages y desplegará los siguientes mensajes:



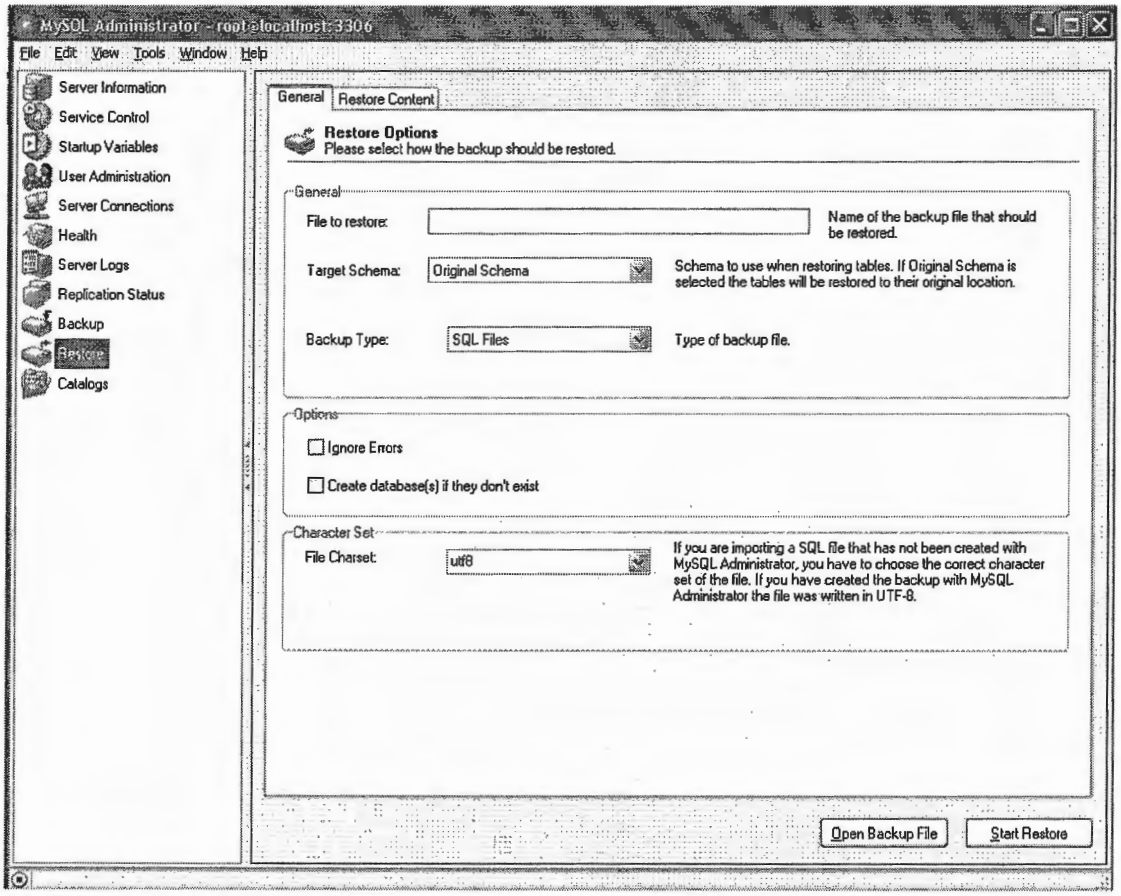

## *13.6. Instalación de la Aplicación de Eigenfaces*

El prototipo de reconocimiento de rostros a partir de Eigenvectores consta de 2 partes, la base de datos en MySQL y los programas script en Python; lo primero que es necesario hacer es restaurar la base de datos que utiliza el programa; para esto se utiliza la herramienta MySQL Administrador instalada en la sección 13.3 de este documento; los pasos se detallan a continuación:

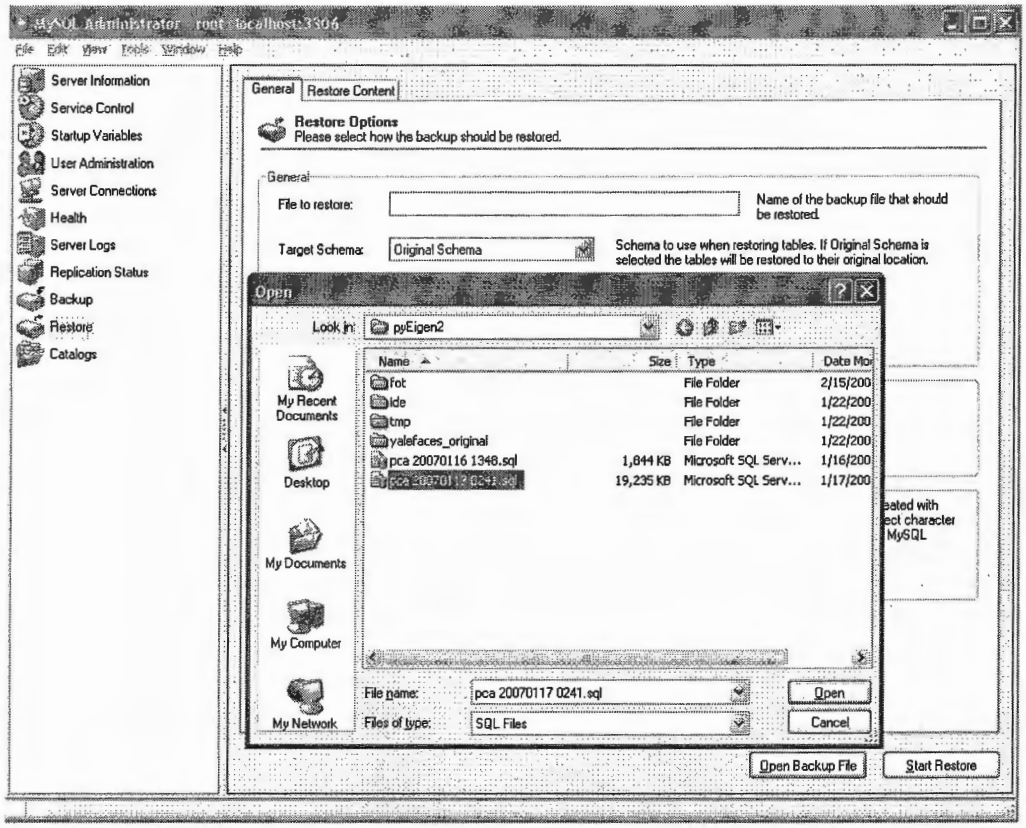
1. Iniciar la herramienta: MySQL Administrador e iniciar sesión con el usuario root y el password correspondiente:

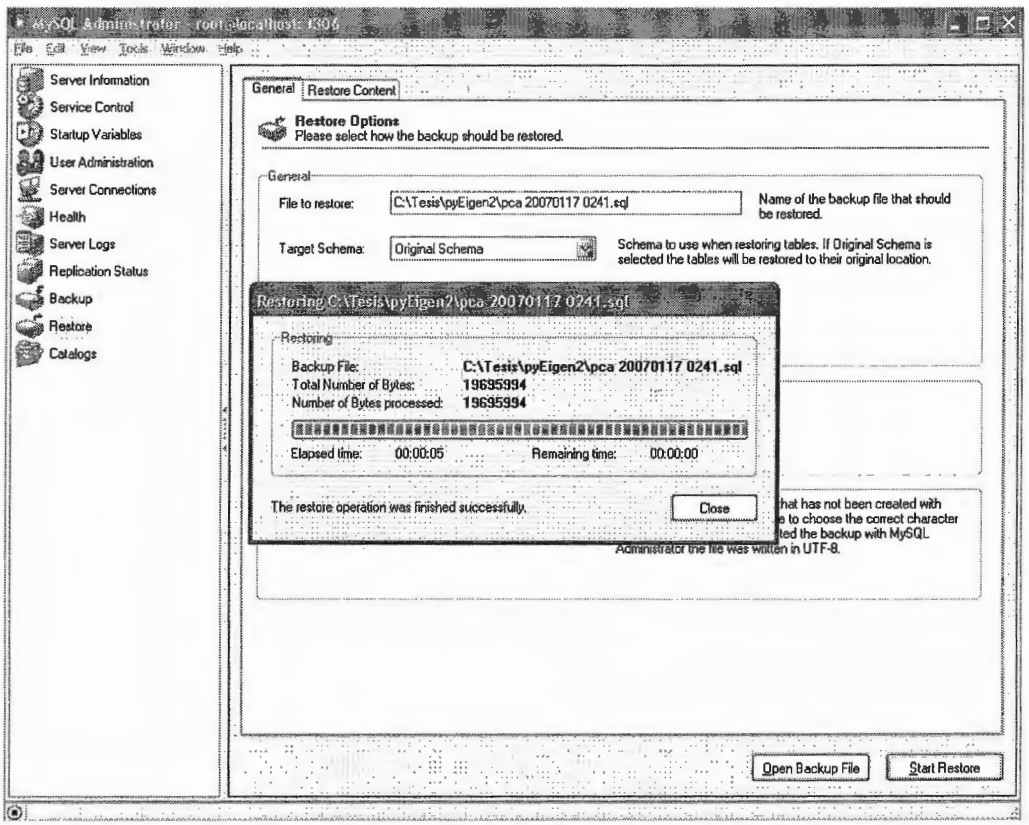


2. Seleccionar la opción Restore de la panel izquierdo del programa, y en el panel derecho seleccionar el boton Open Backup File para abrir el backup realizado:

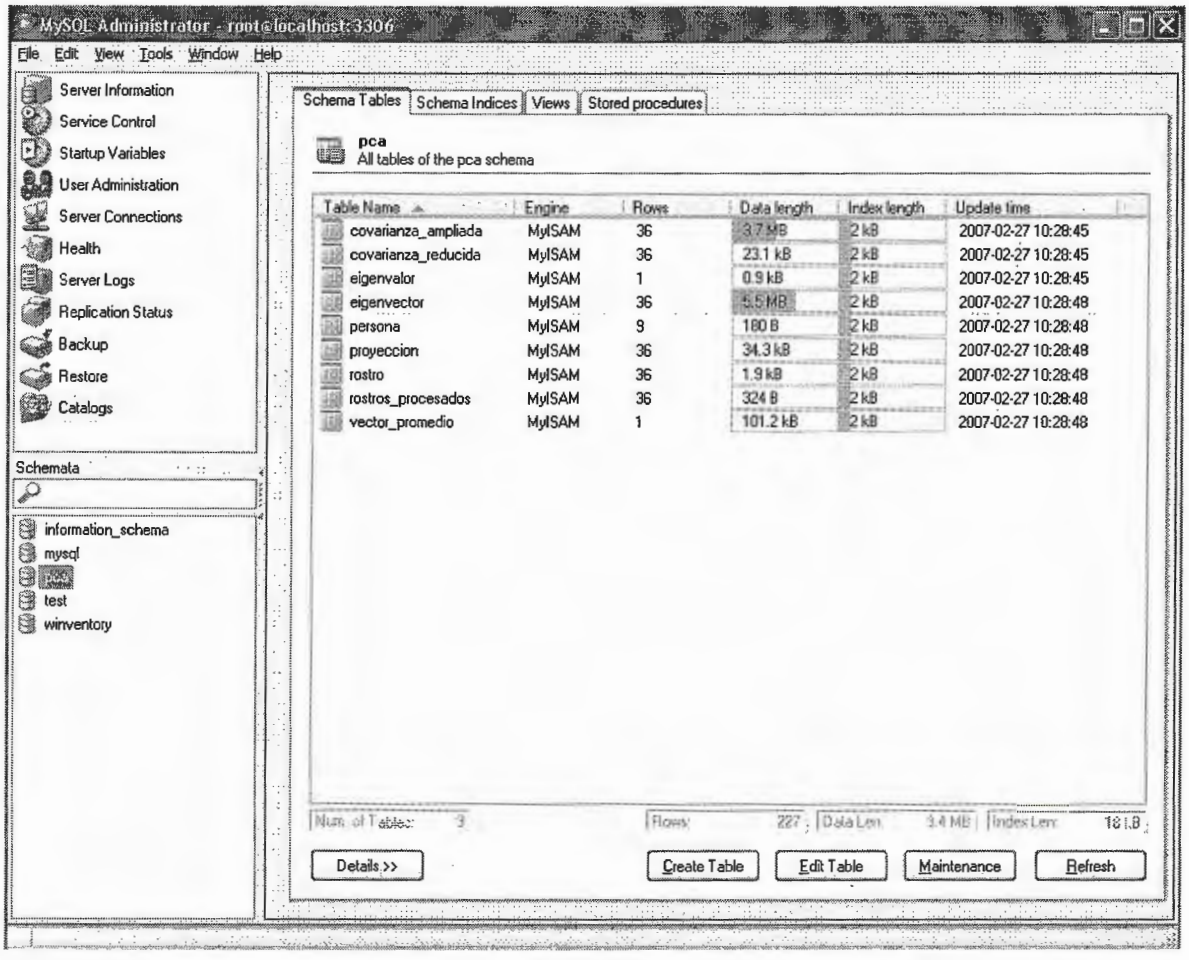



3. MySQL almacena los backup como un script de SQL, los cuales los renombra con el nombre de la base de datos + la fecha del backup; en este caso seleccionamos el backup más reciente y se le da la opción Star Restore.

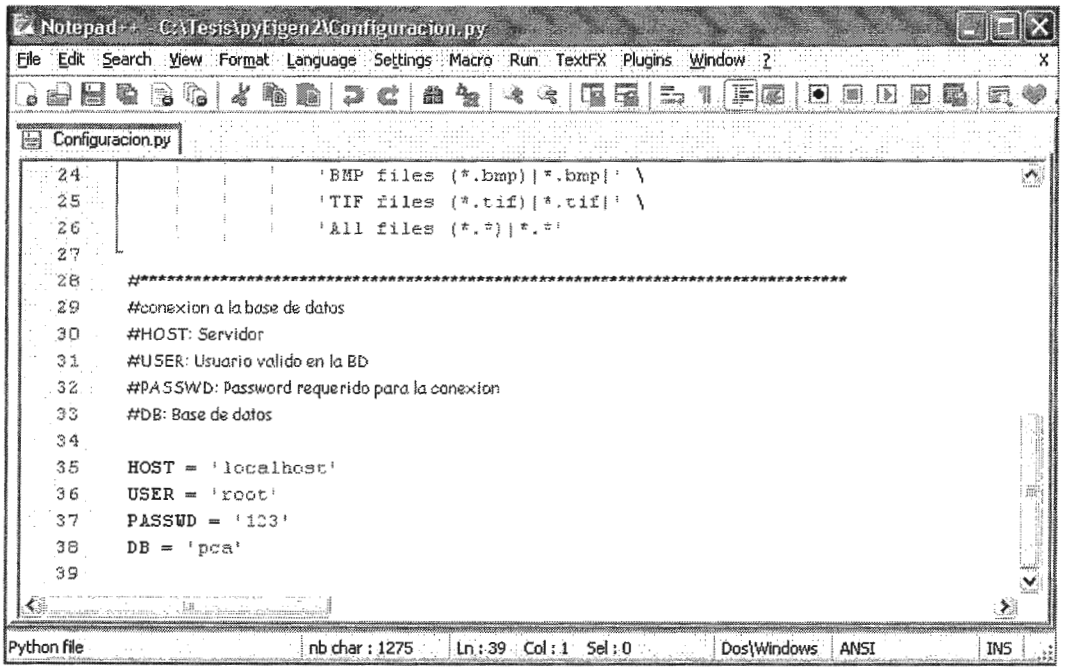

MySQL Administrator - root@localhost:3306
File Edit View Tools Window Help
Server Information
Service Control
Startup Variables
User Administration
Server Connections
Health
Server Logs
Replication Status
Backup
Restore
Catalogs
General
Restore Content
Restore Options
Please select how the backup should be restored.
General
File to restore:
Name of the backup file that should be restored.
Target Schema:
Original Schema
Schema to use when restoring tables. If Original Schema is selected the tables will be restored to their original location.
Open
Look in:
pyEigen2
My Recent Documents
Desktop
My Documents
My Computer
My Network
Name
Size
Type
Date Mo
fot
File Folder
2/15/200
ide
File Folder
1/22/200
tmp
File Folder
1/22/200
yalefaces_original
File Folder
1/22/200
pca 20070116 1348.sql
1,844 KB
Microsoft SQL Serv...
1/16/200
pca 20070117 0241.sql
19,235 KB
Microsoft SQL Serv...
1/17/200
File name:
pca 20070117 0241.sql
Files of type:
SQL Files
Open
Cancel
Open Backup File
Start Restore


MySQL Administrator - root@localhost:3306
File Edit View Tools Window Help
Server Information
Service Control
Startup Variables
User Administration
Server Connections
Health
Server Logs
Replication Status
Backup
Restore
Catalogs
General
Restore Content
Restore Options
Please select how the backup should be restored.
General
File to restore:
C:\Tesis\pyEigen2\pca 20070117 0241.sql
Name of the backup file that should be restored.
Target Schema:
Original Schema
Schema to use when restoring tables. If Original Schema is selected the tables will be restored to their original location.
Restoring C:\Tesis\pyEigen2\pca 20070117 0241.sql
Restoring
Backup File:
C:\Tesis\pyEigen2\pca 20070117 0241.sql
Total Number of Bytes:
19695994
Number of Bytes processed:
19695994
Elapsed time:
00:00:05
Remaining time:
00:00:00
The restore operation was finished successfully.
Close
Open Backup File
Start Restore

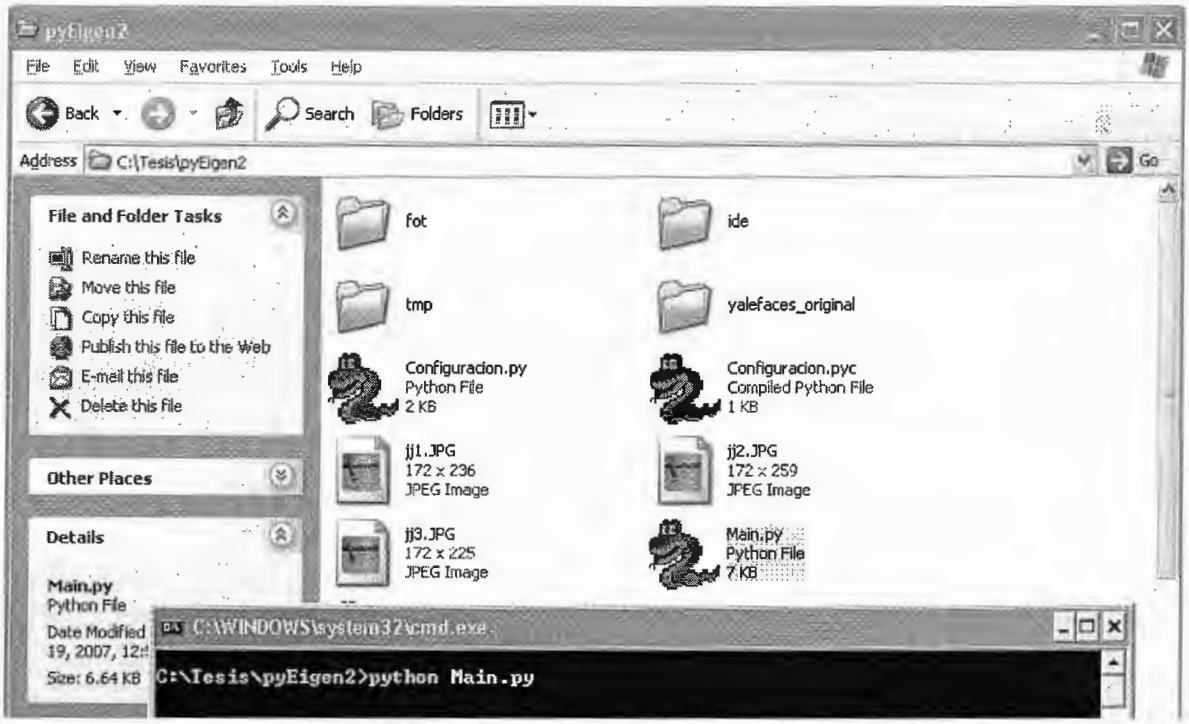
4. Después de la restauración, se habra creado o reemplazado (si existiese previamente) una base de datos denominada PCA, la cual contiene las siguientes tablas:
    - Covarianza_ampliada
    - Covarianza_reducida
    - Eigenvalor
    - Eigenvector
    - Persona
    - Proyeccion
    - Rostro
    - Rostros_procesados
    - Vector_promedio

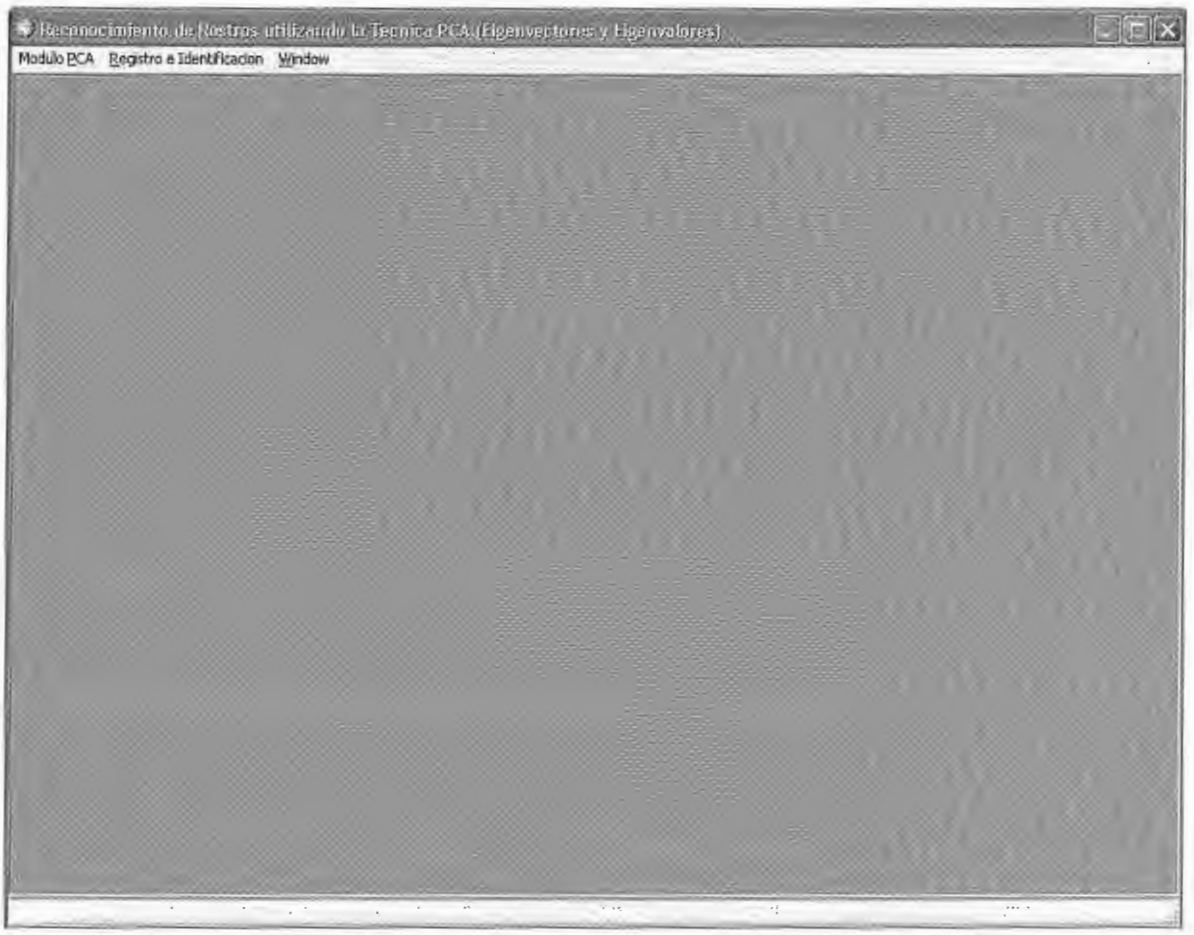


5. Una vez instalada la base de datos; debe copiarse el directorio PyEigen2 a cualquier ubicación en el disco duro; y modificarse el archivo: configuracion.py, el cual contiene las variables:
    - HOST: nombre del servidor que contiene la base de datos.
    - USER: usuario con acceso a la base de datos PCA
    - PASSWD: password para acceder al servidor MySQL
    - DB: nombre de la base de datos, en caso se trabajara con otro nombre diferente a PCA.




6. Una vez realizados todos los pasos anteriores; bastara con dar doble clic al archivo: Main.py o ejecutarlo desde la linea de comandos usando la siguiente instrucción:
    - Python Main.py

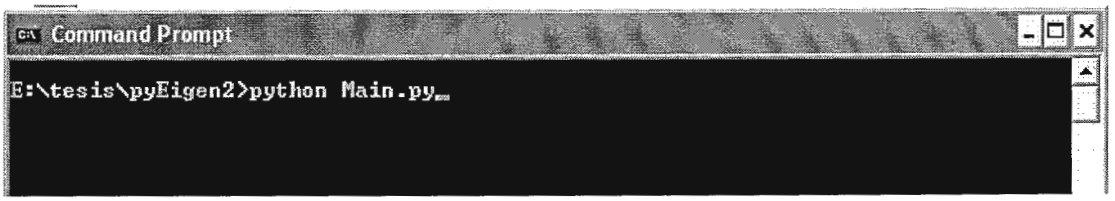

pyEigen2
File Edit View Favorites Tools Help
Back Search Folders
Address C:\Tesis\pyEigen2
Go
File and Folder Tasks
Rename this file
Move this file
Copy this file
Publish this file to the Web
E-mail this file
Delete this file
Other Places
Details
Main.py
Python File
Date Modified
19, 2007, 12:
Size: 6.64 KB
fot
ide
tmp
yalefaces_original
Configuracion.py
Python File
2 KB
Configuracion.pyc
Compiled Python File
1 KB
jj1.JPG
172 x 236
JPEG Image
jj2.JPG
172 x 259
JPEG Image
jj3.JPG
172 x 225
JPEG Image
Main.py
Python File
7 KB
C:\WINDOWS\system32\cmd.exe
C:\Tesis\pyEigen2>python Main.py


Reconocimiento de Rostros utilizando la Tecnica PCA (Eigenvectores y Eigenvalores)
Modulo PCA Registro e Identificacion Window

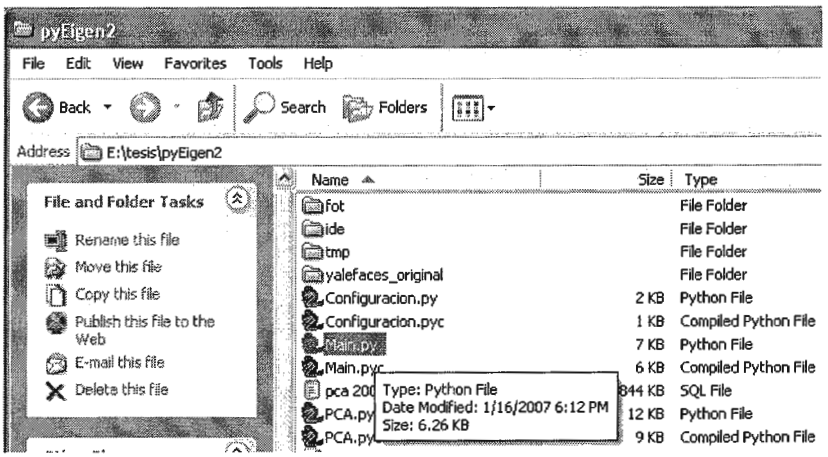
# 14. MANUAL DEL USUARIO PARA LA INTERFAZ GRÁFICA

La ejecución del programa se realiza mediante 2 formas:

- Invocando desde la línea de comando el interprete y el programa principal Main.py



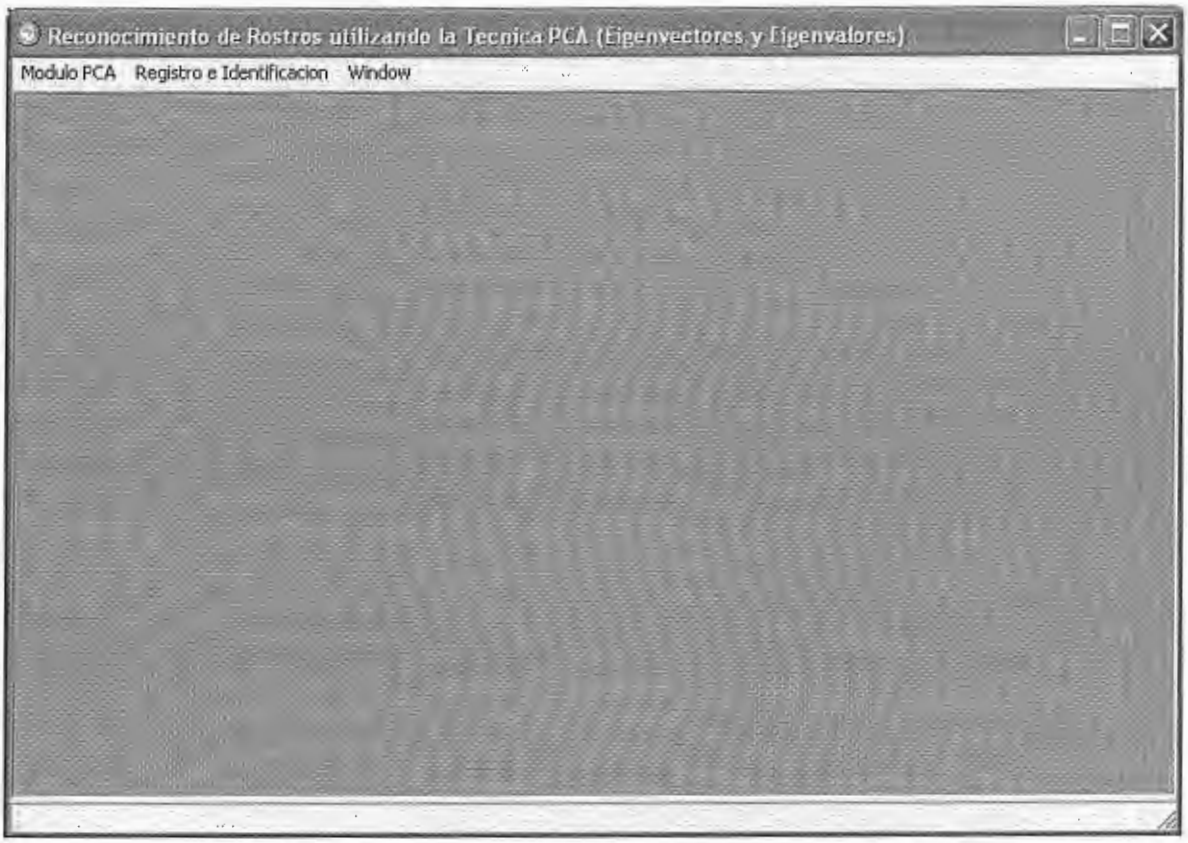

- Haciendo doble clic sobre el archivo Main.py, de esta forma el sistema operativo se encargara automáticamente de buscar el intérprete de Python más adecuado para ejecutar el script.




- Se debe verificar que este ejecutandose el servidor de MySQL, con una base denominada "pca".

La primera vez que se ejecuta el programa probablemente se tardara unos 10 seg. en presentar la pantalla principal, esto se debe a que Python recorre todos los archivos con el objeto de compilarlos aquellas librerías o clases que son implementadas por el programa.

La pantalla principal del prototipo de reconocimiento facial es la siguiente:




El prototipo se encuentra dividio en 2 secciones; las cuales se pueden visualizar en la barra de menú:

- la sección Módulo PCA se emplea cuando se desea entrenar al prototipo para que reconozca los rostros almacenados en la base de datos; está seccion solo es necesaria ejecutarla cuando se registren nuevos rostros y/o personas en el sistema.
- la otra sección es llamada Registro e Identificación; se emplea para registrar a todas aquellas personas y sus rostros digitalizados; así como para la identificación de un rostro desconocido contra todos los rostros registrados en la base de datos.

Los pasos necesarios para entrenar al sistema es registrar una cantidad de personas y asociarles uno o más rostros disponibles, de

preferencia se procura que los rostros abarquen diferentes ángulos de la persona, de esta forma se incrementa la posibilidad de un mejor reconocimiento.

El módulo de registro de personas y rostros se accede a través de la opción: Personas a Identificar:


Se desplegará una forma que permite ingresar los datos básicos de la persona (primer nombre, segundo nombre, primer apellido y segundo apellido) y a través de una interfaz sencilla se pueden asignar y/o visualizar los rostros asociados a cada persona:


El módulo de identificación se accede mediante la opción: Identificar Imagen:


El módulo de identificación de imágenes se encuentra dividido en 3 secciones:

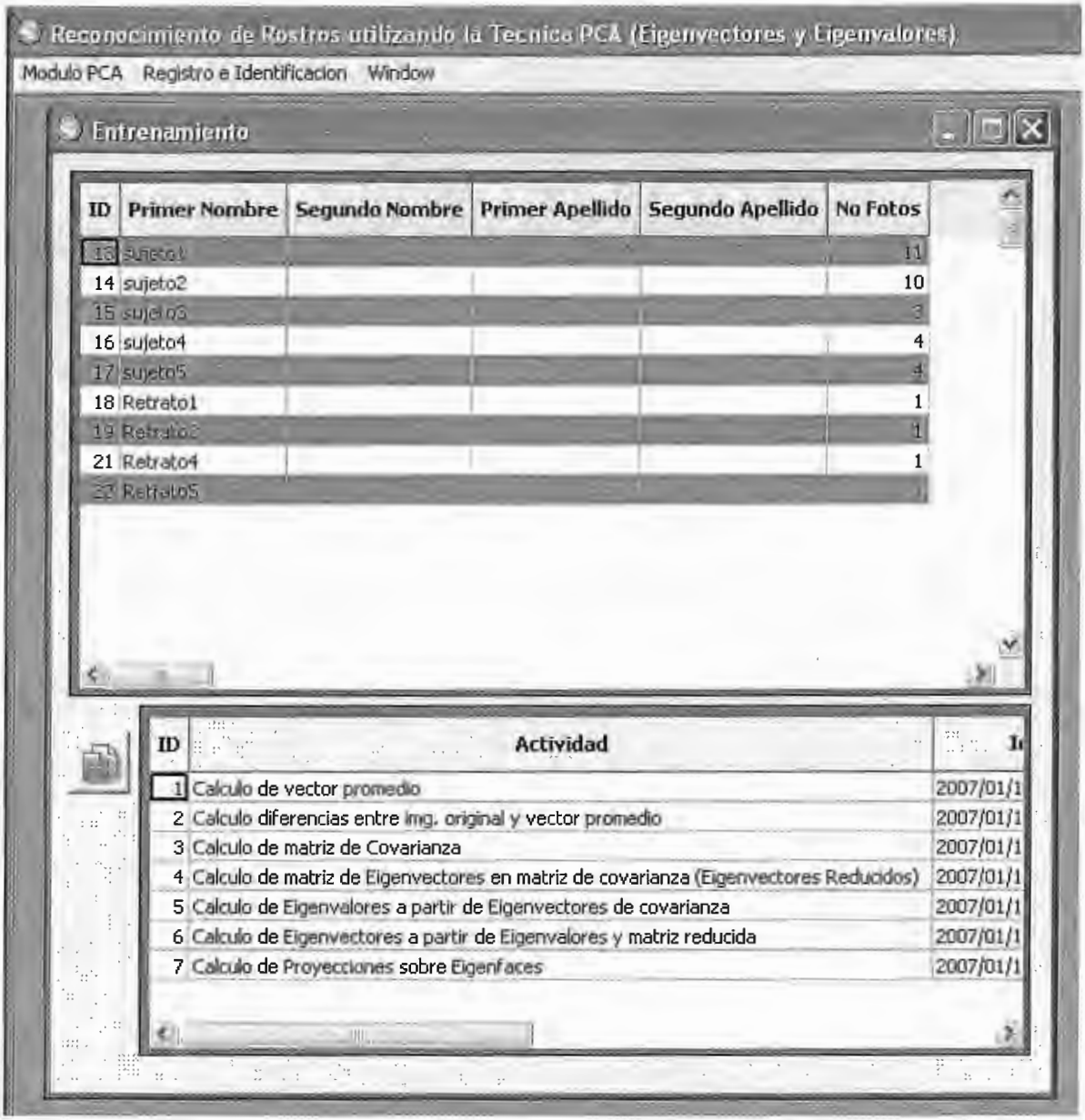
- Sección de la imagen desconocida: puede ser una imagen obtenida de una cámara digital, un retrato hablado, una imagen que presenta gestos o accesorios ausentes en la imagen registrada (por ejemplo, se puede identificar un rostro si esta utilizando anteojos oscuros, si el rostro esta en otra posición o esta haciendo gestos extraordinarios: sonriendo, bostezando, etc.)
- Detalle de similitudes: esta sección se caracteriza por un grid que refleja la fotografía registrada en el sistema que más se aproxima al rostro desconocido, la diferencia euclidea (parámetro utilizado para medir similitudes) y un porcentaje de certeza.
- Sección de la imagen reconstruida: es una imagen en menor tamaño reconstruida a partir del espacio reducido por los eigenvectores y eigenvalores.


Antes de poder identificar cualquier rostro (retrato hablado u rostro en otra posición) es necesario entrenar al sistema; esto se realiza cada vez que se asocian nuevos rostros o personas al sistema; se emplea el módulo: Entrenar.


La pantalla de entrenamiento se encuentra dividia en 2 secciones:

- Resumen de personas registradas en el sistema y cantidad de rostros asociadas.
- Resumen de actividades finalizadas en el entrenamiento; el entrenamiento puede ser tardado dependiendo de la cantidad

de rostros asociados, esto se debe a la cantidad de operaciones matematicas necesarias en el cálculo de los eigenvectores y eigenvalores

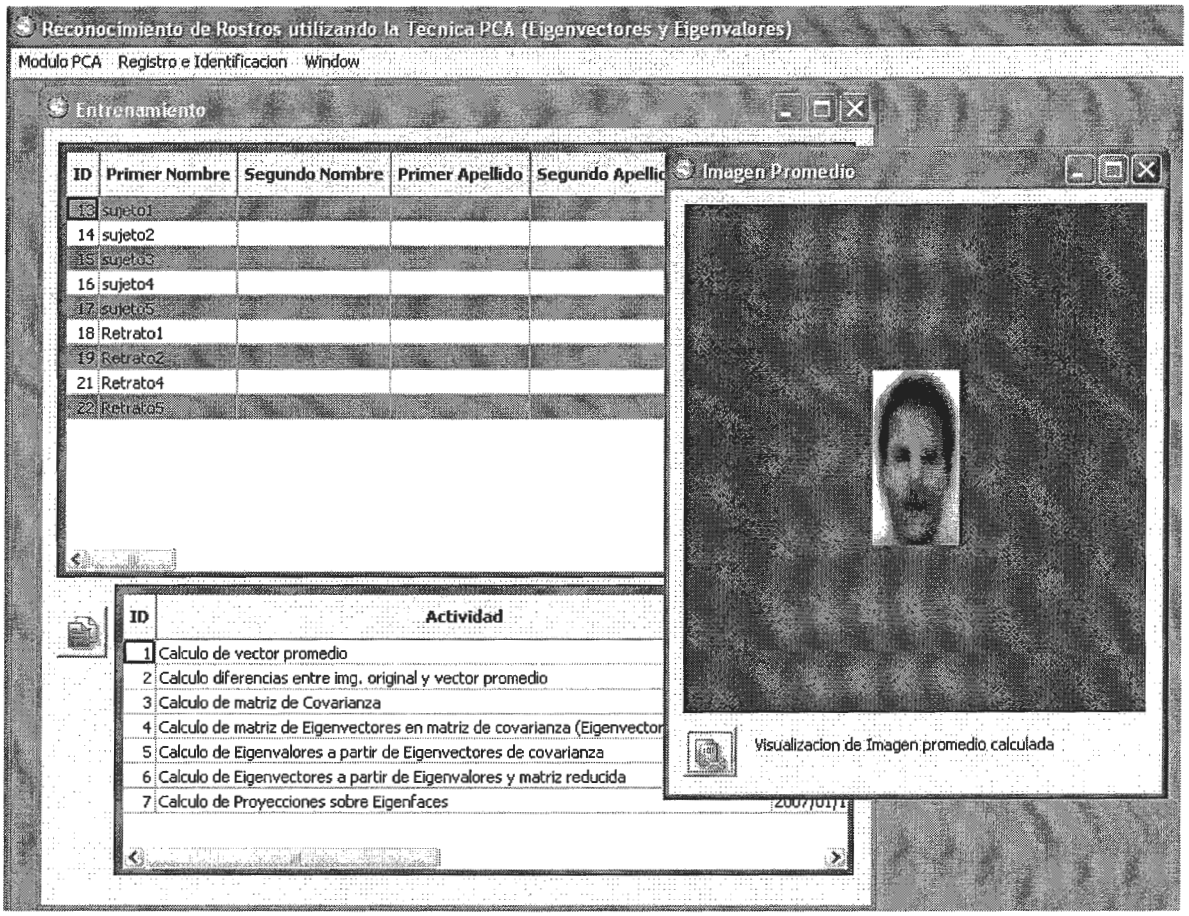



En el módulo PCA se puede observar a nivel detalle los pasos efectuados, hora de inicio, hora de finalizacion y una breve descripción

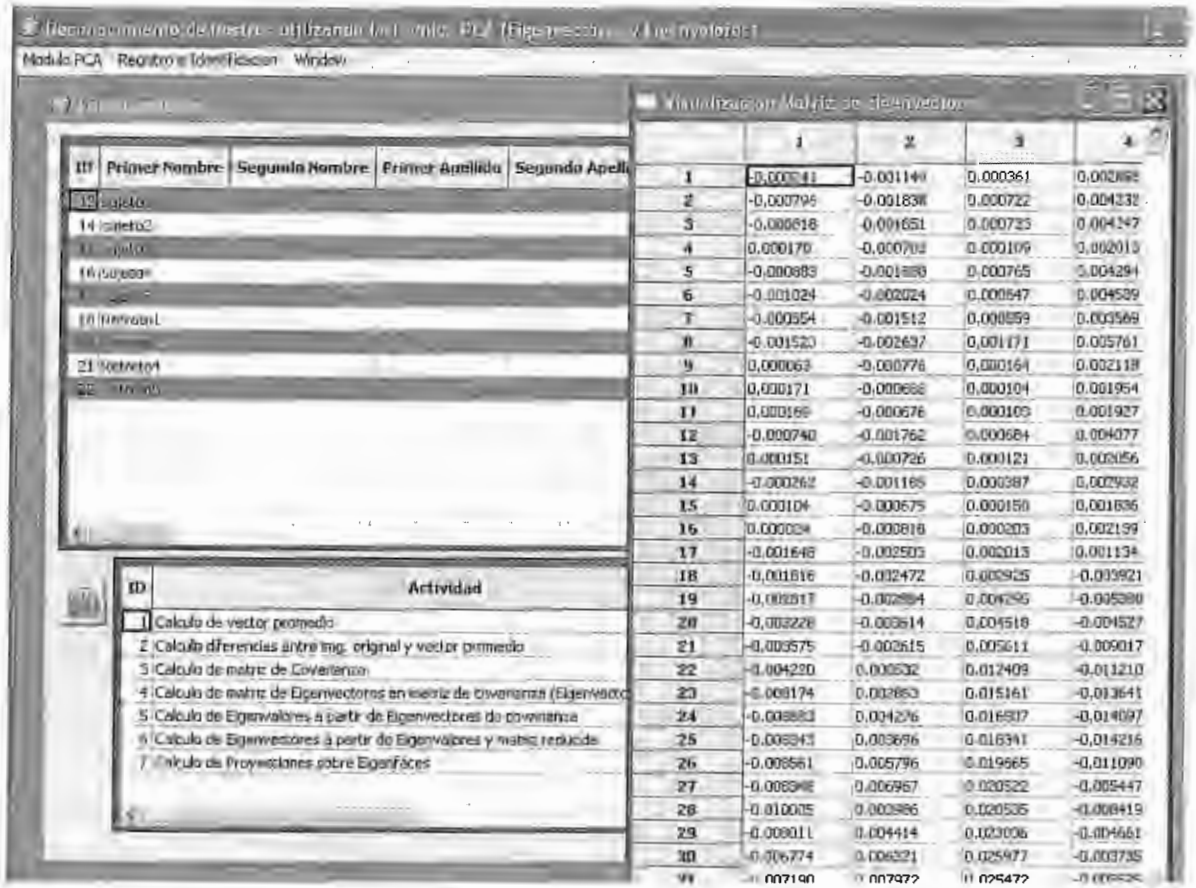
de cada paso (cantidad de archivos procesados, tamaño de matrices calculadas, etc.).

Tambien se puede observar diferentes matrices las cuales son:

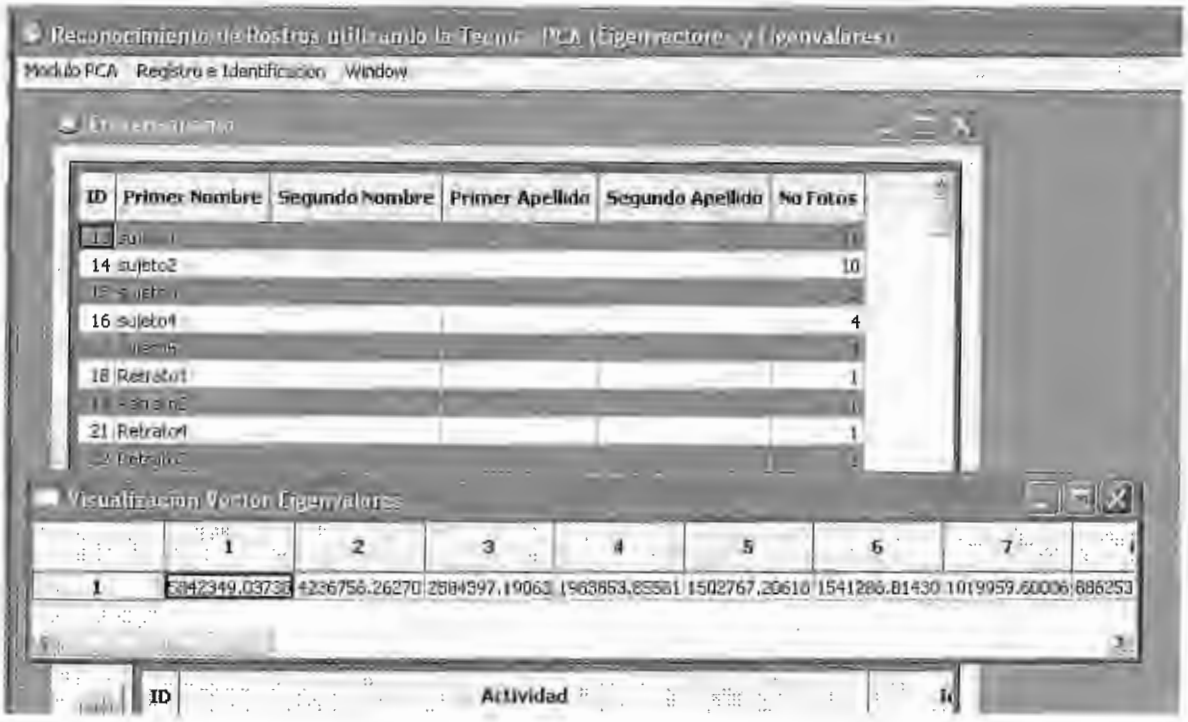
- Imagen promedio del set de entrenamiento.



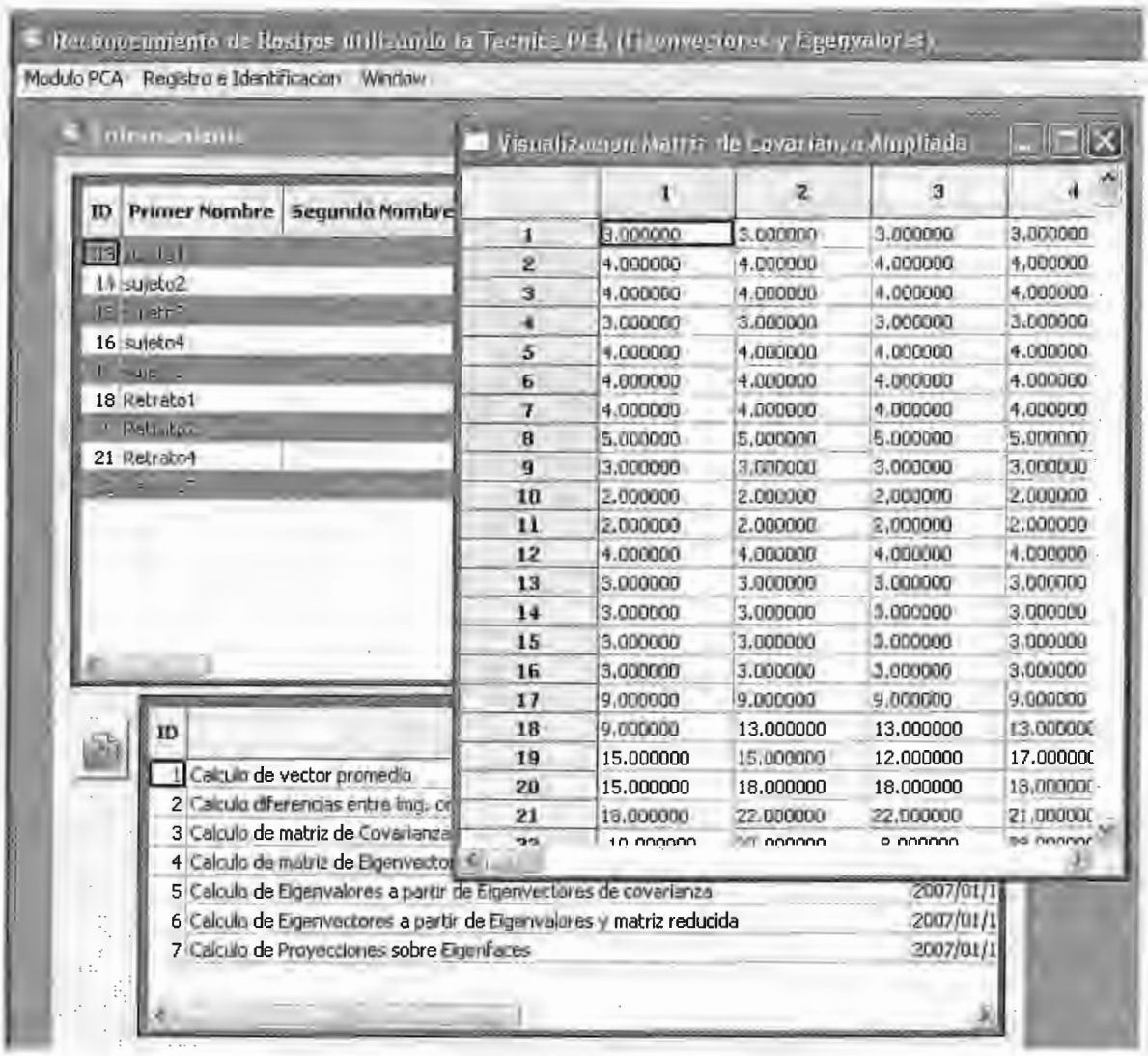

- La matriz de los Eigenvectores.




- El vector de los Eigenvalores

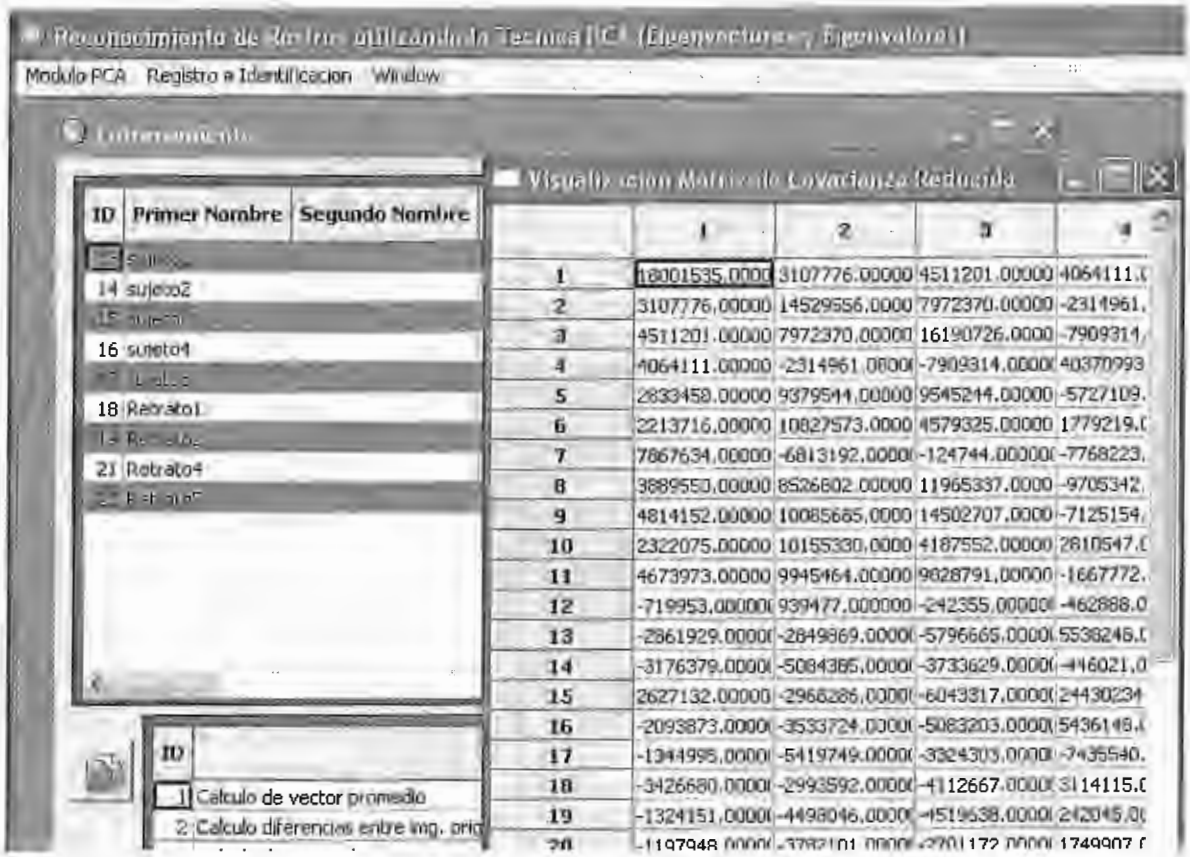


- La Matriz de Covarianza Ampliada

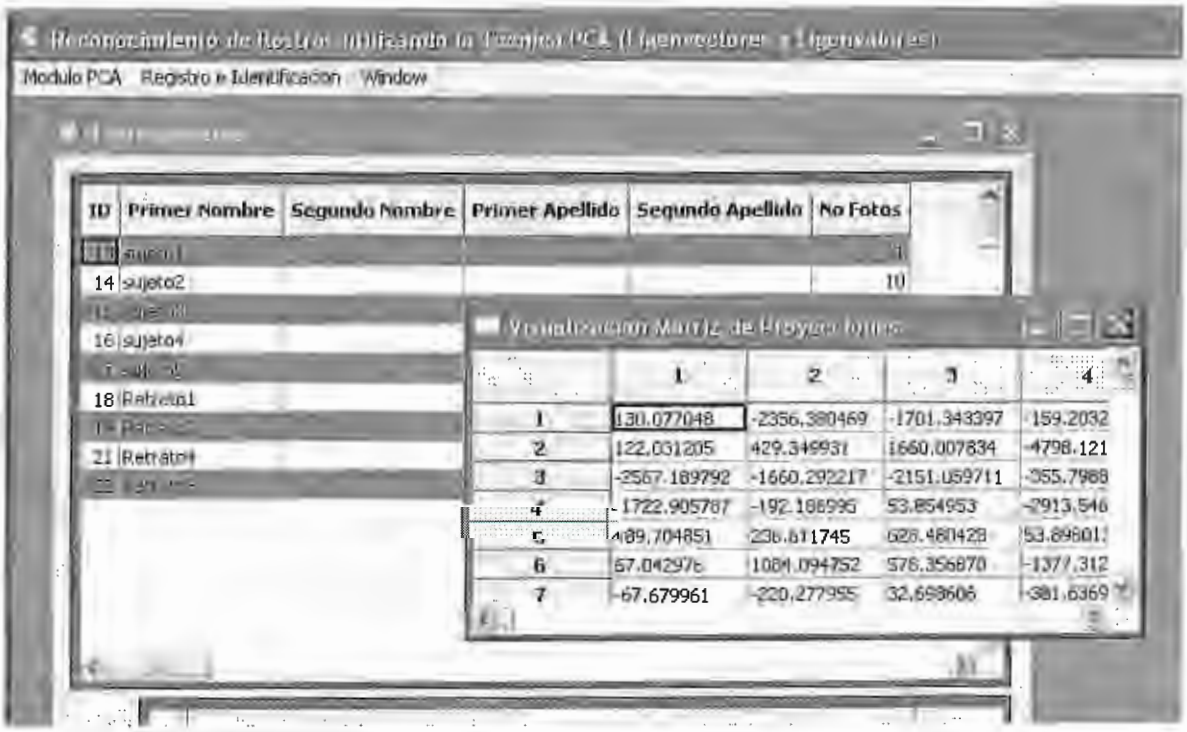



| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 3.000000 | 3.000000 | 3.000000 | 3.000000 |
| 2 | 4.000000 | 4.000000 | 4.000000 | 4.000000 |
| 3 | 4.000000 | 4.000000 | 4.000000 | 4.000000 |
| 4 | 3.000000 | 3.000000 | 3.000000 | 3.000000 |
| 5 | 4.000000 | 4.000000 | 4.000000 | 4.000000 |
| 6 | 4.000000 | 4.000000 | 4.000000 | 4.000000 |
| 7 | 4.000000 | 4.000000 | 4.000000 | 4.000000 |
| 8 | 5.000000 | 5.000000 | 5.000000 | 5.000000 |
| 9 | 3.000000 | 3.000000 | 3.000000 | 3.000000 |
| 10 | 2.000000 | 2.000000 | 2.000000 | 2.000000 |
| 11 | 2.000000 | 2.000000 | 2.000000 | 2.000000 |
| 12 | 4.000000 | 4.000000 | 4.000000 | 4.000000 |
| 13 | 3.000000 | 3.000000 | 3.000000 | 3.000000 |
| 14 | 3.000000 | 3.000000 | 3.000000 | 3.000000 |
| 15 | 3.000000 | 3.000000 | 3.000000 | 3.000000 |
| 16 | 3.000000 | 3.000000 | 3.000000 | 3.000000 |
| 17 | 9.000000 | 9.000000 | 9.000000 | 9.000000 |
| 18 | 9.000000 | 13.000000 | 13.000000 | 13.000000 |
| 19 | 15.000000 | 15.000000 | 12.000000 | 17.000000 |
| 20 | 15.000000 | 18.000000 | 18.000000 | 18.000000 |
| 21 | 18.000000 | 22.000000 | 22.000000 | 21.000000 |

- La matriz de Covarianza Reducida.




- La matriz de las proyecciones



# 15. CONCLUSIONES Y RECOMENDACIONES

A continuación se exponen conclusiones y recomendaciones generadas a partir del estudio analítico y experimental de la metodología PCA; este estudio ha incluído técnicas de extracción de características lineales así como tratamiento de imágenes; en efecto se han cubierto exitosamente todos los objetivos propuestos; tanto específicos como globales.

1. La metodología e investigación en la identificación de rostros humanos propuesta en la tesis, es completamente novedosa en El Salvador, y por tanto su desarrollo se limita al realizado en la presente proyecto; esto hace que su aplicabilidad práctica potencial sea muy grande, pero que por el momento sean necesarios desarrollos adicionales para permitir su implantación en sistemas reales de todo tipo. No obstante, incluso en su estado actual, el prototipo es capaz de funcionar normalmente para cualquier sujeto con el entrenamiento adecuado, esto es, la generación de los modelos de apariencia PCA; el ajuste de las imágenes y el control técnico en la utilización realizado por un experto en materia biométrica.

2. En la presente tesis se han realizado diferentes aportaciones dentro del campo de la investigación y el desarrollo del reconocimiento visual de características automatizadas. En conjunto, se ha propuesto un método de identificación de rostros humanos partiendo de la teoría PCA, capaz de operar entornos controlados semiestructurados. El desarrollo de este sistema ha requerido analizar en profundidad las arquitecturas

de reconocimiento basado en apariencia (reconocimiento con intervención humana) y las técnicas de extracción de características principales. El estudio de este campo ha dado lugar a diversas aportaciones de carácter experimental, científico y pedagógico, las cuales han sido, plasmadas a lo largo de todo el documento. Una de las principales aportaciones de esta tesis es la realización de un estudio analítico y experimental de las similitudes y diferencias de los métodos de extracción de características mas utilizados en la actualidad: PCA e ICA

3. El presente proyecto ha propuesto un prototipo basado en el método PCA, que cumple una serie de requisitos o cualidades que le validan como una prometedora investigación para el desarrollo de un Sistema Biométrico de identificación de rostros. Entre las características más relevantes del prototipo en miras a su expansión, se pueden destacar las siguientes:
    a. *Capacidad de trabajar con objetos de forma libre*, debido al uso de características multicanales locales obtenidas a partir de su apariencia.
    b. *Aplicabilidad en entornos no estructurados*, puesto que los modelos del objeto que se emplean presentan invariancia ante cambios de iluminación y pose del objeto ante la cámara
    c. *Posibilidad de trabajar a múltiples escalas*, esto es, la deteccion del objeto puede realizarse aun cuando el objeto no esta siempre a una distancia de la cámara conocida y establecida

4. El desempeño del prototipo desarrollado en el lenguaje interpretado Pyhon es aceptable para bases de rostros inferiores a 500 imágenes, donde se estima que el tiempo de entrenamiento sería aproximadamente de 5.5 minutos; y el tiempo de reconocimiento aproximadamente 2.5 minutos; estos tiempos mejorarían implementando el prototipo en un lenguaje que genere código máquina; C/C++ sería el lenguaje idoneo si se busca el más alto desempeño, sin embargo debido a la complejidad se podría optar por el lenguaje D [16] el cual maneja estructuras de alto nivel (similares a las que se encuentran en Python) y administra la memoria eficientemente; el código generado en C/C++ o D se ejecutaría aproximadamente 13 veces más rápido.

5. El reconocimiento de rostros utilizando la técnica PCA si bien demanda procesamiento y almacenamiento de las matrices de proyecciones relativamente grandes (por ejemplo, el procesamiento de 500 imágenes de 60x120 pixeles implica generar una matriz del tamaño: 7200 x 500 es decir: 3,600,000 elementos) sin embargo el reconocimiento es muy efectivo; el método provee mucha tolerancia para reconocer rostros de personas aun si utilizan accesorios, las condiciones de luz son diferentes, o las expresiones faciales son muy diferentes a los rostros registrados inicialmente.

6. El reconocimiento de rostros a través de los Eigenvectores y Eigenvalores se utiliza con mucha frecuencia para localizar o

---

[16] Lenguaje D: http://www.digitalmars.com/d/

Benchmarks: http://shootout.alioth.debian.org/debian/benchmark.php?test=knucleotide&lang=all

efectuar búsquedas de personas siempre y cuando se posee una imagen digital (obtenida a través de cámaras de vigilancia) lo cual hace lo hace un método eficaz de monitoreo en zonas públicas para la búsqueda de criminales y/o terroristas; sin embargo en este proyecto se tuvo la oportunidad de observar el comportamiento de identificación partiendo de rostros hablados generados por el software "Faces" otorgados por la Division Técnica y Científica de la Policía Nacional Civil; se puede concluir que la búsqueda de sospechosos partiendo de retratos hablados es válida ya que en todas las pruebas el método PCA posicionó a la persona buscada a través del retrato hablado entre los primeros 4 candidatos.

7. El reconocimiento de rostros a partir de retratos hablados sería mas eficiente si se desarrolla el módulo editor de retratos hablados tipificado según rasgos de personas latinas; de preferencia personas del área de CentroAmérica; el desarrollo de este módulo debe ser considerado como otro trabajo de graduación debido al tiempo que involucra tomar fotografias de una muestra representativa; el desarrollo del módulo puede utilizar, según disposición de las partes involucradas, el presente documento de tesis como base para la investigación y desarrollo del prototipo.

8. El presente prototipo puede implementarse en un ambiente empresarial, sin embargo es recomendable se considere evaluar un gestor de base de datos de acorde al volumen de la información a registrar, asi como considerar opciones de clustering para enfrentar la complejidad computacional implicada en los algoritmos del núcleo.

9. El prototipo desarrollado en la presente tesis deja abierto el camino para multitud de trabajos futuros, entre los que cabe destacar: Prototipos de Selección e Identificación de Características utilizando el método ICA, o una combinación PCA-ICA; ambos métodos han demostrado una gran fortaleza cuando son utilizados como métodos de comprensión de la información, a continuación se enumeran una serie de trabajos relacionados los cuales podrían ser valiosas investigaciones y desarrollos a futuro:
    a. *Utilización de todas las componentes y aplicación de clasificadores invariantes ante rotaciones*: En el caso de realizar selección de características, los resultados obtenidos por PCA e ICA pueden ser diferentes. Se plantea como trabajo futuro el estudio tanto experimental como teórico de posibles procedimientos de selección de características con ICA (con PCA la selección mas idónea suele ser la elección de componentes de máxima a mínima varianza, salvo algunas excepciones en el caso de aplicaciones de reconocimiento de objetos donde las clases son muy cercanas, como en el reconocimiento de caras, donde se suelen descartar las 3 o 4 primeras componentes y mantener el resto ordenadas de máxima a mínima varianza). Actualmente no existen criterios que permitan determinar cuales de las componentes independientes consiguen una mejor separación entre clases; el objetivo es, por una parte, buscar criterios plausibles; y por otra, determinar si la selección de características sobre ICA permite mejorar los resultados

obtenidos al realizar selección de características sobre PCA.

b. Implementación en paralelo. Es posible paralelizar múltiples procesamientos, de forma que el tiempo de cómputo se puede reducir apreciablemente. Como trabajo futuro se considera la implementación del detector en un sistema multiprocesador o bien en un entorno de red. Caben dos estrategias de paralelización:
    i. Procesar cada canal o grupo de canales y cada escala o grupo de escalas en procesadores diferentes para posteriormente promediar los resultados obtenidos; ó
    ii. Procesar cada ventana o grupo de ventanas en equipos diferentes, para luego combinar los resultados correspondientes a cada píxel. El resultado sería un sistema de identificación que permitiría compaginar un alto nivel de precisión (paso pequeño entre ventanas y muchas escalas) con una velocidad de procesado muy elevada (funcionamiento en tiempo real).
c. *Aplicación a problemas de categorización.* Otro posible trabajo futuro sobre los desarrollos de la tesis es la aplicación a problemas de categorización o catalogación, en los cuales el objetivo no es detectar un cierto objeto específico sino detectar cualquier objeto perteneciente a una cierta clase. En principio, el método presentado en la tesis seria aplicable directamente a este tipo de problemas, y solo sería necesario modificar el tipo de imágenes de entrenamiento suministradas (en lugar de múltiples vistas de un cierto objeto, múltiples vistas de

varios objetos pertenecientes a una clase determinada, podría ser una solución). La principal diferencia con el problema de detección es la variación en la influencia o peso de los canales o señales

d. *Extracción del contorno de los objetos detectados.* Para un detector de objetos desarrollado según la técnica PCA, puede ser necesario determinar el contorno de los objetos detectados. Por ejemplo, para un brazo robot que agarre objetos, no basta con determinar la zona donde se encuentra el objeto en la escena, sino que se necesita conocer el contorno exterior del mismo. En este sentido, se ha iniciado una línea de trabajo basada en la utilización de modelos deformables o *snakes.* Tales modelos se inicializan con la forma de la región devuelta por el detector de objetos (unión de píxeles cuyo valor de probabilidad supera un determinado umbral) y que se deforma de acuerdo con una función potencial que tiende a aproximar el contorno a los bordes presentes en la imagen (extraídos con un detector de bordes estandar como por ejemplo el detector de Canny)

# 16. BIBLIOGRAFIA


- http://es.wikipedia.org/wiki/Biometr%C3%ADa (Marzo 28, 2006)
- http://www.face-rec.org (Marzo 29, 2006)
- http://mathworld.wolfram.com/Eigenvector.html (Abril 5, 2006)
- Fonseca Rosales, Joaquín Armando; Orellana Crespín, Mario Ernesto; Rivera García, Juan Carlos; Tesis: Prototipo de Software de Reconocimiento Óptico del Alfabeto Internacional del Lenguaje de Sordos; Universidad Don Bosco, San Salvador, El Salvador; Septiembre 2005.
- Turk, Matthew A. y Pentland, Alex P.; Face Recognition Using Eigenrostros; Vision and Modeling Group, The Media Laboratory Massachusetts Institute of Technology.
- Pentland Alex, Moghaddam Baback, Starner Thad; View-Based and Modular Eigenspaces for Face Recognition; IEEE Conference on Computer Vision & Pattern Recognition, 1994.
- Menser B., Müller F.; Face Detection in Color Images Using Principal Components Analysis, Aachen University of Technology (RWTH), Germany.
- Seul, Michael; O'Gorman, Lawrence; Sammon, Michael J.; Practical Algorithms for Image Analysis, Description, Examples and Code; Cambridge University Press, ISBN 0-521-66065-3, 2005. www.cambridge.org
- Maltoni, Davide; Maio, Dario; Jain, Anil K.; Prabhakar Salil; Handbook of Fingerprint Recognition; Springer Professional Computer, ISBN 0-387-95431-7, 2003. www.springeronline.com
- Liu, Chengjun; Wechsler Harry; Comparative Assessment of Independent Component Analysis (ICA) for Face Recognition; George Mason University, Department of Computer Science 1999.

- Haag Michael, Romberg Justin, Meza Fara, Jackson Erika; Algebra Lineal: Conceptos Básicos. Documento producido para el proyecto The Connexions, bajo la licencia de Creative Commons Attribution.
- Daniels, Doug; Cox, Steven; Vector Space. Documento producido para el proyecto The Connexions, bajo la licencia de Creative Commons Attribution.
- Daniela, Doug; Cox, Steven; Subspaces. Documento producido para el proyecto The Connexions, bajo la licencia de Creative Commons Attribution
- Vicente, M. A.; Reinoso, O.; Pérez, C.; Sabater, J.A.; Azorín, J.A.; Reconocimiento de Objetos 3D mediante Análisis PCA; Universidad Miguel Hernández, División Ingeniería de Sistemas y Automática, Departamento de Ingeniería, Campus de Elche; Alicante, España.
- Haag, Michael; Romberg, Justin; Meza, Fara; Jackson, Erika; Eigenvectores y Eigenvalores; Documento producido para el proyecto The Connexions, bajo la licencia de Creative Commons Attributions.
- Schmuller, Joseph; Aprendiendo UML en 24 horas; Prentice Hall Ebook.

# 17. ANEXOS

## *17.1. Código Fuente*

### 17.1.1. Archivo: Main.py

```
import sys
import Configuracion
sys.path.append(Configuracion.dirApp)

import ide
import ide.Entrenar
import ide.Grid
import ide.Identificar
import ide.MntPersona
import ide.BD
import ide.images
import ide.ImagenPromedio
import wx
# libreria PCA
import PCA

ID_Exit = wx.NewId()
ID_Entrenar  = wx.NewId()
ID_ImagenPromedio = wx.NewId()
ID_GridEigenvector = wx.NewId()
ID_GridCovarianzaAmp = wx.NewId()
ID_GridCovarianzaRed = wx.NewId()
ID_GridProyecciones = wx.NewId()
ID_GridEigenvalor = wx.NewId()

# Identificacion y Registro
ID_PersonaMnt = wx.NewId()
ID_Identificar = wx.NewId()

# objeto Rostro
rostro = ide.BD.Rostro()
# objeto Vectores (recuperar vectores de la BD)
vector = ide.BD.Vectores()

class ParentFrame(wx.MDIParentFrame):
   def __init__(self):
      wx.MDIParentFrame.__init__(self,
```

```
        None,-1,
        'Reconocimiento de Rostros utilizando la Tecnica PCA (Eigenvectores y Eigenvalores)',
        size=(900,700))

    self._icon = ide.images.getpythonIcon()
    self.SetIcon(self._icon)

    self.bg_bmp = ide.images.getGridBGBitmap()
    self.GetClientWindow().Bind(
        wx.EVT_ERASE_BACKGROUND, self.OnEraseBackground
    )


    menu = wx.Menu()
    menu.Append(ID_Entrenar,'&Entrenar')
    menu.AppendSeparator()
    menu.Append(ID_ImagenPromedio,'Imagen Promedio')
    menu.Append(ID_GridEigenvector,'Grid E&igenvector')
    menu.Append(ID_GridEigenvalor,'Grid Eigen&valor')
    menu.Append(ID_GridCovarianzaAmp,'Grid Covarianza &Ampliada')
    menu.Append(ID_GridCovarianzaRed,'Grid &Covarianza Reducida')
    menu.Append(ID_GridProyecciones,'Grid de &Proyecciones')
    menu.AppendSeparator()
    menu.Append(ID_Exit,'E&xit')

    menubar = wx.MenuBar()
    menubar.Append(menu,'Modulo &PCA')

    menu = wx.Menu()
    menu.Append(ID_PersonaMnt,'Personas a identificar')
    menu.Append(ID_Identificar,'I&dentificar Imagen')
    menubar.Append(menu,'&Registro e Identificacion')

    self.SetMenuBar(menubar)

    self.CreateStatusBar()

    self.Bind(wx.EVT_MENU,self.OnEntrenar,id=ID_Entrenar)
    self.Bind(wx.EVT_MENU,self.OnImagenPromedio,id=ID_ImagenPromedio)
    self.Bind(wx.EVT_MENU,self.OnGridEigenvector,id=ID_GridEigenvector)
    self.Bind(wx.EVT_MENU,self.OnGridEigenvalor,id=ID_GridEigenvalor)
    self.Bind(wx.EVT_MENU,self.OnGridCovarianzaAmp,id=ID_GridCovarianzaAmp)
    self.Bind(wx.EVT_MENU,self.OnGridCovarianzaRed,id=ID_GridCovarianzaRed)
    self.Bind(wx.EVT_MENU,self.OnGridProyecciones,id=ID_GridProyecciones)
```

```
        self.Bind(wx.EVT_MENU,self.OnExit,id=ID_Exit)

        self.Bind(wx.EVT_MENU,self.OnPersonaMnt,id=ID_PersonaMnt)
        self.Bind(wx.EVT_MENU,self.OnIdentificar,id=ID_Identificar)

    def OnEraseBackground(self, evt):
        dc = evt.GetDC()
        if not dc:
            dc = wx.ClientDC(self.GetClientWindow())
        sz = self.GetClientSize()
        w = self.bg_bmp.GetWidth()
        h = self.bg_bmp.GetHeight()
        x = 0
        while x < sz.width:
            y = 0
            while y < sz.height:
                dc.DrawBitmap(self.bg_bmp, x, y)
                y = y + h
            x = x + w

    def OnExit(self,evt):
        self.Close(True)

    def OnEntrenar(self,evt):
        win = ide.Entrenar.Frame(self,-1,'Entrenamiento',size=(580,570),
                                 style = wx.DEFAULT_FRAME_STYLE)
        win.Show(True)

    def OnIdentificar(self,evt):
        #win = ide.Identificar.Frame(self,-1,"Identificar Imagen",klt,PCA,vp,size=(750,400),
                                 #style=wx.DEFAULT_FRAME_STYLE)
        win = ide.Identificar.Frame(self,-1,'Identificar Rostro',size=(890,400),
                 style=wx.DEFAULT_FRAME_STYLE)
        win.Show(True)

    def OnImagenPromedio(self,evt):
        win = ide.ImagenPromedio.Frame(self,-1,'Imagen Promedio',size=(350,450),
                 style=wx.DEFAULT_FRAME_STYLE)
        win.Show(True)

    def OnGridEigenvector(self,evt):
        # recuperando de la BD para mostrar en grid
        eigenvector = vector.EigenvectorRecuperarMatriz()
        win = ide.Grid.Frame(self,-1,'Visualizacion Matriz de Eigenvectores',eigenvector,
```

```
            size=(400,200),style=wx.DEFAULT_FRAME_STYLE)
        win.Show(True)

    def OnGridEigenvalor(self,evt):
        # recuperando de la BD los eigenvalores
        eigenvalor = vector.EigenvalorRecuperar()
        win = ide.Grid.Frame(self,-1,'Visualizacion Vector Eigenvalores',eigenvalor,
                size=(300,200),style=wx.DEFAULT_FRAME_STYLE)
        win.Show(True)

    def OnGridCovarianzaAmp(self,evt):
        # recuperando de la BD la covarianza ampliada
        covarianzaAmpliada = vector.CovarianzaAmpliadaRecuperarMatriz()
        win = ide.Grid.Frame(self,-1,'Visualizacion Matriz de Covarianza Ampliada',
                covarianzaAmpliada,size=(400,200),style=wx.DEFAULT_FRAME_STYLE)
        win.Show(True)

    def OnGridCovarianzaRed(self,evt):
        # recuperando de la BD la covarianza reducida
        covarianzaReducida = vector.CovarianzaReducidaRecuperarMatriz()
        win = ide.Grid.Frame(self,-1,'Visualizacion Matriz de Covarianza Reducida',
                covarianzaReducida,size=(400,200),style=wx.DEFAULT_FRAME_STYLE)
        win.Show(True)

    def OnGridProyecciones(self,evt):
        # recuperando de la BD las proyecciones
        proyeccion = vector.ProyeccionRecuperarMatriz()
        win = ide.Grid.Frame(self,-1,'Visualizacion Matriz de Proyecciones',
                proyeccion,size=(400,200),style=wx.DEFAULT_FRAME_STYLE)
        win.Show(True)

    def OnPersonaMnt(self,evt):
        win = ide.MntPersona.Frame(self,-1,'Personas',size=(890,450),style=wx.DEFAULT_FRAME_STYLE)
        win.Show(True)

if __name__ == '__main__':
    class MyApp(wx.App):
        def OnInit(self):
            wx.InitAllImageHandlers()
            frame = ParentFrame()
            frame.Show(True)
            self.SetTopWindow(frame)
            return True
```

```
app = MyApp(False)
app.MainLoop()
```

## 17.1.2. Archivo: Configuración.py

```
import os
import os.path
import wx

#*****************************************************************************
#ubicacion de la aplicacion
dirApp = os.getcwd()

#*****************************************************************************
#ubicacion de archivos temporales
dirTmp = os.path.join(dirApp,'tmp')

#*****************************************************************************
#ubicacion de archivos de interfaz grafica y recursos para botones, etc.
dirIde = os.path.join(dirApp,'ide')
dirRes = os.path.join(dirApp,'res')
dirFot = os.path.join(dirApp,'fot')

#*****************************************************************************
#tipos de archivos soportados
wildcard = 'PNG Files (*.png)|*.png|' \
           'JPG Files (*.jpg)|*.jpg|' \
           'GIF files (*.gif)|*.gif|' \
           'BMP files (*.bmp)|*.bmp|' \
           'TIF files (*.tif)|*.tif|' \
           'All files (*.*)|*.*'

#*****************************************************************************
#conexion a la base de datos
#HOST: Servidor
#USER: Usuario valido en la BD
#PASSWD: Password requerido para la conexion
#DB: Base de datos

HOST = 'localhost'
USER = 'root'
PASSWD = '123'
```

```
DB = 'pca'
```

## 17.1.3. Archivo: PCA.py

```
# libreria PIL (Python Imaging Library)
import Image

# libreria numpy para objetos array
import numpy

# libreria linalg para calculo de eigenvectores y eigenvalores
import numpy.linalg as linalg

# libreria scipy para interfaz con PIL (guardar imagenes y transformar imagenes en array)
import scipy.misc.pilutil as pilutil

# atributo de modulo, indica la dimension original de las imagenes a tratar
# es empleado cuando se redimensiona los vectores al tamano original
dim_org = (120,60)

# funcion que lee un archivo de imagen y lo retorna en un vector (insumo para las
# clases VectorPromedio y KLT
def ArchivoImagenAVector(archivo):
    i = Image.open(archivo)
    i = i.resize((dim_org[1],dim_org[0]))
    matrizImagen = pilutil.fromimage(i)
    return matrizImagen.reshape((matrizImagen.size,1))

# funcion que convierte la imagen vectorizada en un archivo de imagen
# utilizada por clases VectorPromedio y KLT
def VectorAArchivoImagen(archivo,vector):
    vector = vector.reshape(dim_org)
    pilutil.imsave(archivo,vector)

def VectorAArchivoTxt(archivo,vector):
        f = open('./tmp/' + archivo,'w')

        if len(vector.shape)==2:
            x,y = vector.shape
            for i in range(x):
                for j in range(y):
                    f.write(str(vector[i][j])+'\t')
```

```
            f.write('\n')
        else:
            x = vector.shape[0]
            for i in range(x):
                f.write(str(vector[i])+'\n')

        f.close()

class VectorPromedio:
    def __init__(self):
        # dim: dimension del vector (tupla)
        v = numpy.array(range(dim_org[0]*dim_org[1]))
        # convirtiendo el array en un vector nulo
        self._vector_suma = v.reshape((v.size,1)) * 0
        self._n = 0

    def Promediar(self,vector):
        # sumamos matrices y registramos cuantas matrices se han procesado
        self._vector_suma = self._vector_suma + vector
        self._n += 1

    def Promedio(self):
        # retorna la suma total de las matrices entre el numero procesados (vector)

        #escribiendo la matriz en un archivo de texto para depuracion
        #VectorAArchivoTxt('vector_promedio.txt',self._vector_suma/self._n)

        return self._vector_suma / self._n

    def Diferencia(self,vector):
        # retorna la matriz de diferencia entre el vector y el promedio
        return vector - self.Promedio()

# transformada de Karhunen-Loeve
class KLT:
    def __init__(self):
        dim = dim_org[0]*dim_org[1]

        self._A = numpy.array(range(dim))
        # convirtiendo el array en un vector nulo
        self._A = self._A.reshape((self._A.size,1)) * 0
        self._n = 0
        self._Aid = []
```

```
        # vector de eigenvectors
        self._eigenvector = numpy.array(range(dim))
        self._eigenvector = self._eigenvector.reshape((self._eigenvector.size,1))*0.00
        self._n_eigenvector = 0

        # vector de eigenvalores
        self._eigenvalor = 0

        # matriz de covarianza
        self._C = numpy.zeros([1])
        self._C = self._C.reshape((self._C.size,1))

        # vector de proyecciones de otra imagen sobre los eigenvectores
        self._proy_img = numpy.zeros([1])

        # vector de proyecciones de las imagenes originales
        self._proyeccion = numpy.zeros([1])
        self._proyeccion = self._proyeccion.reshape((self._proyeccion.size,1))

        # vector eigenvectores reducidos
        self._EVecRed = numpy.zeros([1])
        # vector eigenvalores reducidos
        self._EValRed = numpy.zeros([1])

        # vector de distancias euclideas entre proyecciones originales y proyeccion de imagen desconocida
        self._euclidea = numpy.zeros([1])

    def ConcatenarVector(self,vector,id):
        # el vector A es la concatenacion de las imagenes - la imagen promedio
        if self._n==0:
            #self._A = vector.astype(numpy.Float)
            #self._A = vector.astype(float)
            self._A = vector
            self._Aid.append(id)
        else:
            self._A = numpy.concatenate((self._A,vector),1)
            self._Aid.append(id)
        self._n += 1

    def CalcularCovarianza(self):
        # la covarianza se calcula como la multiplicacion de transpuesta
        # de la matriz A por si misma, dividida entre el no. de vectores
        #self._C = numpy.matrixmultiply(numpy.transpose(self._A),self._A)
        self._C = numpy.dot(numpy.transpose(self._A),self._A)
```

```
    #escribiendo la matriz en un archivo de texto para su depuracion
    #VectorAArchivoTxt('covarianza.txt',self._C)

  def CalcularEigenRed(self):
    # calcula los eigenvectores y eigenvalores de la matriz
    # de covarianza reducida
    #self._EValRed, self._EVecRed = linalg.eigenvectors(self._C)
    self._EValRed, self._EVecRed = linalg.eig(self._C)
    self._EVecRed = numpy.transpose(self._EVecRed)

    #escribiendo las matrices en archivos de texto para su depuracion
    #VectorAArchivoTxt('eigenvectores_red.txt',self._EVecRed)
    #VectorAArchivoTxt('eigenvalores_red.txt',self._EValRed)


#**************************************************************************
#calculo de Eigenvalores y Eigenvectores definitivos
def CalcularEigenvalores(self):
    self._eigenvalor = 1.0/self._n * self._EValRed

    #escribiendo las matrices en archivos de texto para su depuracion
    #VectorAArchivoTxt('eigenvalores.txt',self._eigenvalor)

def CalcularEigenvectores(self):
    for i in range(self._n):
        eigenvector = self.CalcularEigenvector(i)
        if self._n_eigenvector==0:
            self._eigenvector = eigenvector
        else:
            self._eigenvector = numpy.concatenate(
                (self._eigenvector,eigenvector),1)
        self._n_eigenvector += 1

    #escribiendo las matrices en archivos de texto para su depuracion
    #VectorAArchivoTxt('eigenvector.txt',self._eigenvector)

def CalcularEigenvector(self,i):
    val = numpy.sqrt(self._EValRed[i])
    vec = self._EVecRed[i]
    #res = 1.0/val*numpy.matrixmultiply(self._A,vec)
    res = 1.0/val*numpy.dot(self._A,vec)
    return res.reshape((res.size,1))
```

```
    print numpy.dot(numpy.transpose(res),res)

#*****************************************************************

def CalcularProyecciones(self):
    # calcula todas las proyecciones de las imagenes con media nula sobre los eigenvectores
    #self._proyeccion = numpy.array(range(self._n_eigenvector*self._n),numpy.Float)
    self._proyeccion = numpy.array(range(self._n_eigenvector*self._n),float)
    self._proyeccion = self._proyeccion.reshape((self._n,self._n_eigenvector))
    for i in range(self._n):
        for j in range(self._n_eigenvector):
            phi = numpy.transpose(self._A)[j]
            phi = phi.reshape((1,phi.size))
            u = numpy.transpose(self._eigenvector)[i]
            u = u.reshape((u.size,1))
            #self._proyeccion[i][j] = numpy.matrixmultiply(phi,u)
            self._proyeccion[i][j] = numpy.dot(phi,u)

    #escribiendo las matrices en archivos de texto para su depuracion
    #VectorAArchivoTxt('proyeccion.txt',self._proyeccion)

def Eigenface(self,i):
    eigenface = numpy.transpose(self._eigenvector)[i]
    return eigenface.reshape((eigenface.size,1))

def SecuenciaEigenvaluesRepr(self,n):
    # genera la secuencia de los eigenvalues mas representativos
    # la cantidad de eigenvalues es indicada por el parametro n
    sec = []
    orden_asc = numpy.argsort(self._eigenvalor)
    orden_asc = orden_asc[0]
    for i in range(n):
        j = orden_asc[orden_asc.size-1-i]
        sec.append(j)
    return sec

def ReconstruirImagen(self,vp,j,n):
    # reconstruye la j-esima imagen utilizando n eigenvectores representativos
    if (n>self._n): n=self._n

    sec = self.SecuenciaEigenvaluesRepr(n)

    for i in sec:
        ui = self.Eigenface(i)
```

```
            wij = self._proyeccion[i][j]
            vp = vp + wij*ui
        return vp

    def CalcularProyeccionesImagen_r(self,vector,vectorPromedio):
        self._proy_img = numpy.array(range(self._n_eigenvector),float)*0
        for i in range(self._n_eigenvector):
            ut = numpy.transpose(self._eigenvector)[i]
            ut = ut.reshape((1,ut.size))
            #self._proy_img[i] = numpy.matrixmultiply(ut,vector-vectorPromedio)
            self._proy_img[i] = numpy.dot(ut,vector-vectorPromedio)

        #escribiendo las matrices en archivos de texto para su depuracion
        #VectorAArchivoTxt('proyeccion_img.txt',self._proy_img)

    def CalcularProyeccionesImagen(self,vector,vectorPromedio):
        # calcula todas las proyecciones de las imagenes con media nula sobre los eigenvectores
        #self._proyeccion = numpy.array(range(self._n_eigenvector*self._n),numpy.Float)
        self._proy_img = numpy.array(range(self._n_eigenvector),float)*0
        self._proyeccion = self._proyeccion.reshape((self._n,self._n_eigenvector))
        for j in range(self._n_eigenvector):
            phi = vector-vectorPromedio
            phi = phi.reshape((1,phi.size))
            u = numpy.transpose(self._eigenvector)[j]
            u = u.reshape((u.size,1))
            #self._proyeccion[i][j] = numpy.matrixmultiply(phi,u)
            self._proy_img[j] = numpy.dot(phi,u)

    def CalcularDistanciaEuclidea_r(self):
        # calcula la distancia euclidea existente entre:
        # - el vector self._proyeccion (proyecciones de imagenes originales)
        # - el vector self._proy_img (proyeccion de imagen desconocida)
        self._euclidea = numpy.array(range(self._n),float)*0
        for j in range(self._n_eigenvector):
            d_euclidea = 0
            for i in range(self._n):
                d_euclidea += (self._proy_img[i] - self._proyeccion[i][j])**2
            self._euclidea[j] = d_euclidea

        #escribiendo las matrices en archivos de texto para su depuracion
        #VectorAArchivoTxt('euclidea.txt',self._euclidea)

    def CalcularDistanciaEuclidea(self):
        # calcula la distancia euclidea existente entre:
```

```
        # - el vector self._proyeccion (proyecciones de imagenes originales)
        # - el vector self._proy_img (proyeccion de imagen desconocida)
        self._euclidea = numpy.array(range(self._n),float)*0
        for i in range(self._n):
            d_euclidea = 0
            for j in range(self._n_eigenvector):
                d_euclidea += (self._proy_img[j] - self._proyeccion[j][i])**2
            self._euclidea[i] = d_euclidea

        #escribiendo las matrices en archivos de texto para su depuracion
        #VectorAArchivoTxt('euclidea.txt',self._euclidea)

    def IdentificacionDistanciaEuclidea(self):
        return numpy.argsort(self._euclidea)

    def MatrizDistancia(self):
        #forma la matriz de distancias, ordenada de forma descendente
        # la matriz consta de 3 columnas:
        # columna correlativo
        # columna id
        # columna distancia

        ide = self.IdentificacionDistanciaEuclidea()
        filas = ide.size
        matriz = numpy.array(range(filas*3),float)
        matriz = matriz.reshape((filas,3))

        mayor = 0.

        for i in range(filas):
            #matriz[i][0] = self._Aid[ide[i]]
            matriz[i][0] = ide[i]
            matriz[i][1] = self._euclidea[ide[i]]
            matriz[i][2] = 0
            if (mayor < self._euclidea[ide[i]]): mayor = self._euclidea[ide[i]]

        for i in range(filas):
            matriz[i][2] = (mayor-matriz[i][1])/mayor*100.

        return matriz


if __name__ == "__main__":
print 'hi'
```

## 17.1.4. Archivo: ide/__init__.py

```
# Paquete para Interfaz Grafica
# para aplicacion de Reconocimiento de Rostros
# a traves de la tecnica PCA (Eigenvectores y Eigenvalores)
```

## 17.1.5. Archivo: ide/BD.py

```
import adodb
import Configuracion
import pickle
import os
import numpy

class RostroProcesado:
    def __init__(self):
        self._cn = adodb.NewADOConnection('mysql')

    def Limpiar(self):
        self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)
        sql = 'truncate table rostros_procesados'
        self._cn.Execute(sql)
        self._cn.Close()

    def Registrar(self,orden,idRostro):
        self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)
        sql = 'insert into rostros_procesados values(%d,%d)' % (orden,idRostro)
        self._cn.Execute(sql)
        self._cn.Close()

    def Persona(self,orden):
        self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)
        sql = 'select idRostro from rostros_procesados where id=%d' % orden
        idRostro = self._cn.GetOne(sql)
        sql = 'select idPersona from rostro where id=%d' % idRostro
        idPersona = self._cn.GetOne(sql)

        sql = 'select id, nombre1, nombre2, apellido1, apellido2 from persona where id=%d' % idPersona
        cur = self._cn.Execute(sql)
        while not cur.EOF:
            row = cur.fields
            break

        cur.Close()
```

```
        self._cn.Close()
        return row

class Vectores:
    def __init__(self):
        self._cn = adodb.NewADOConnection('mysql')

    # ******************************************************************************
    # seccion eigenvalor
    def EigenvalorGuardar(self,vector):
        self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)

        vector = vector.reshape((1,vector.size))
        # truncando la tabla
        sql = 'truncate table eigenvalor'
        self._cn.Execute(sql)

        # escribiendo el id del vector
        sql = 'insert into eigenvalor(id) values(1)'
        self._cn.Execute(sql)

        #serializando el vector a un archivo temporal
        archivoTmp = os.path.join(Configuracion.dirTmp,'eigenvalor.txt')
        pickle.dump(vector,open(archivoTmp,'wb'))

        #subiendo el archivo al campo longblob
        self._cn.UpdateBlobFile('eigenvalor','vector',archivoTmp,'id=1')

        self._cn.Close()

    def EigenvalorRecuperar(self):
        self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)

        vector = self._cn.GetOne('select vector from eigenvalor where id=1')
        archivoTmp = os.path.join(Configuracion.dirTmp,'eigenvalor.txt')
        f = open(archivoTmp,'wb')
        f.write(vector)
        f.close()

        self._cn.Close()
        v = pickle.load(open(archivoTmp))
        return v
```

```
# ************************************************************************
# seccion vector promedio

def VPromedioGuardar(self,vector):
    self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)

    vector = vector.reshape((1,vector.size))

    # truncando la tabla
    sql = 'truncate table vector_promedio'
    self._cn.Execute(sql)

    # escribiendo el id del vector
    sql = 'insert into vector_promedio(id) values(1)'
    self._cn.Execute(sql)

    #serializando el vector en un archivo temporal
    archivoTmp = os.path.join(Configuracion.dirTmp,'archivopromedio.txt')
    pickle.dump(vector,open(archivoTmp,'wb'))

    #subiendo el archivo al campo longblob
    self._cn.UpdateBlobFile('vector_promedio','vector',archivoTmp,'id=1')

    self._cn.Close()

def VPromedioRecuperar(self):
    self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)

    vector = self._cn.GetOne('select vector from vector_promedio where id=1')
    archivoTmp = os.path.join(Configuracion.dirTmp,'archivopromedio.txt')
    f = open(archivoTmp,'wb')
    f.write(vector)
    f.close()

    self._cn.Close()
    v = pickle.load(open(archivoTmp))
    v = v.reshape((v.size,1))
    return v

# ************************************************************************
# seccion Eigenvector

def EigenvectorId(self):
    self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)
```

```
        resultado = []
        sql = 'select id from eigenvector'
        cur = self._cn.Execute(sql)
        while not cur.EOF:
            row = cur.fields
            resultado.append(row[0])
            cur.MoveNext()
        self._cn.Close()
        return resultado

    def EigenvectorGuardar(self,vector):
        self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)

        sql = 'insert into eigenvector(cor) values(0)'
        cur = self._cn.Execute(sql)
        id = cur.Insert_ID()

        #serializando el vector en un archivo temporal
        archivoTmp = os.path.join(Configuracion.dirTmp,'eigenvector.txt')
        pickle.dump(vector,open(archivoTmp,'wb'))

        #subiendo el archivo al campo longblob
        self._cn.UpdateBlobFile('eigenvector','vector',archivoTmp,'id=%d' % id)

        self._cn.Close()

    def EigenvectorRecuperar(self,id):
        self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)

        vector = self._cn.GetOne('select vector from eigenvector where id=%d' % id)
        archivoTmp = os.path.join(Configuracion.dirTmp,'eigenvector.txt')
        f = open(archivoTmp,'wb')
        f.write(vector)
        f.close()

        self._cn.Close()
        v = pickle.load(open(archivoTmp))
        return v

    def EigenvectorLimpiar(self):
        self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)
        sql = 'truncate table eigenvector'
        self._cn.Execute(sql)
        self._cn.Close()
```

```
def EigenvectorGuardarMatriz(self,matriz):
    self.EigenvectorLimpiar()
    matriz = numpy.transpose(matriz)
    x,y = matriz.shape
    for i in range(x):
        vector = matriz[i]
        self.EigenvectorGuardar(vector)

def EigenvectorRecuperarMatriz(self):
    matriz = None
    ei = self.EigenvectorId()
    for i in ei:
        v = self.EigenvectorRecuperar(i)
        v = v.reshape((v.size,1))
        if matriz==None:
            matriz = v
        else:
            matriz = numpy.concatenate((matriz,v),1)
    return matriz

# ************************************************************************
# seccion matriz covarianza ampliada
def CovarianzaAmpliadaId(self):
    self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)
    resultado = []
    sql = 'select id from covarianza_ampliada'
    cur = self._cn.Execute(sql)
    while not cur.EOF:
        row = cur.fields
        resultado.append(row[0])
        cur.MoveNext()
    self._cn.Close()
    return resultado

def CovarianzaAmpliadaGuardar(self,vector):
    self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)

    sql = 'insert into covarianza_ampliada(cor) values(0)'
    cur = self._cn.Execute(sql)
    id = cur.Insert_ID()

    #serializando el vector en un archivo temporal
    archivoTmp = os.path.join(Configuracion.dirTmp,'covarianza_ampliada.txt')
```

```
        pickle.dump(vector,open(archivoTmp,'wb'))

        #subiendo el archivo al campo longblob
        self._cn.UpdateBlobFile('covarianza_ampliada','vector',archivoTmp,'id=%d' % id)

        self._cn.Close()

    def CovarianzaAmpliadaRecuperar(self,id):
        self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)

        vector = self._cn.GetOne('select vector from covarianza_ampliada where id=%d' % id)
        archivoTmp = os.path.join(Configuracion.dirTmp,'covarianza_ampliada.txt')
        f = open(archivoTmp,'wb')
        f.write(vector)
        f.close()

        self._cn.Close()
        v = pickle.load(open(archivoTmp))
        return v

    def CovarianzaAmpliadaLimpiar(self):
        self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)
        sql = 'truncate table covarianza_ampliada'
        self._cn.Execute(sql)
        self._cn.Close()

    def CovarianzaAmpliadaGuardarMatriz(self,matriz):
        self.CovarianzaAmpliadaLimpiar()
        matriz = numpy.transpose(matriz)
        x,y = matriz.shape
        for i in range(x):
            vector = matriz[i]
            self.CovarianzaAmpliadaGuardar(vector)

    def CovarianzaAmpliadaRecuperarMatriz(self):
        matriz = None
        ca = self.CovarianzaAmpliadaId()
        for i in ca:
            v = self.CovarianzaAmpliadaRecuperar(i)
            v = v.reshape((v.size,1))
            if matriz==None:
                matriz = v
            else:
                matriz = numpy.concatenate((matriz,v),1)
```

```
        return matriz

    # ************************************************************************
    # seccion matriz covarianza reducida
    def CovarianzaReducidaId(self):
        self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)
        resultado = []
        sql = 'select id from covarianza_reducida'
        cur = self._cn.Execute(sql)
        while not cur.EOF:
            row = cur.fields
            resultado.append(row[0])
            cur.MoveNext()
        self._cn.Close()
        return resultado

    def CovarianzaReducidaGuardar(self,vector):
        self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)

        sql = 'insert into covarianza_reducida(cor) values(0)'
        cur = self._cn.Execute(sql)
        id = cur.Insert_ID()

        #serializando el vector en un archivo temporal
        archivoTmp = os.path.join(Configuracion.dirTmp,'covarianza_reducida.txt')
        pickle.dump(vector,open(archivoTmp,'wb'))

        #subiendo el archivo al campo longblob
        self._cn.UpdateBlobFile('covarianza_reducida','vector',archivoTmp,'id=%d' % id)

        self._cn.Close()

    def CovarianzaReducidaRecuperar(self,id):
        self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)

        vector = self._cn.GetOne('select vector from covarianza_reducida where id=%d' % id)
        archivoTmp = os.path.join(Configuracion.dirTmp,'covarianza_reducida.txt')
        f = open(archivoTmp,'wb')
        f.write(vector)
        f.close()

        self._cn.Close()
        v = pickle.load(open(archivoTmp))
        return v
```

```
def CovarianzaReducidaLimpiar(self):
    self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)
    sql = 'truncate table covarianza_reducida'
    self._cn.Execute(sql)
    self._cn.Close()

def CovarianzaReducidaGuardarMatriz(self,matriz):
    self.CovarianzaReducidaLimpiar()
    matriz = numpy.transpose(matriz)
    x,y = matriz.shape
    for i in range(x):
        vector = matriz[i]
        self.CovarianzaReducidaGuardar(vector)

def CovarianzaReducidaRecuperarMatriz(self):
    matriz = None
    cr = self.CovarianzaReducidaId()
    for i in cr:
        v = self.CovarianzaReducidaRecuperar(i)
        v = v.reshape((v.size,1))
        if matriz==None:
            matriz = v
        else:
            matriz = numpy.concatenate((matriz,v),1)
    return matriz

# ************************************************************************
# seccion proyecciones
def ProyeccionId(self):
    self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)
    resultado = []
    sql = 'select id from proyeccion'
    cur = self._cn.Execute(sql)
    while not cur.EOF:
        row = cur.fields
        resultado.append(row[0])
        cur.MoveNext()
    self._cn.Close()
    return resultado

def ProyeccionGuardar(self,vector):
    self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)
```

```
        sql = 'insert into proyeccion(cor) values(0)'
        cur = self._cn.Execute(sql)
        id = cur.Insert_ID()

        #serializando el vector en un archivo temporal
        archivoTmp = os.path.join(Configuracion.dirTmp,'proyeccion.txt')
        pickle.dump(vector,open(archivoTmp,'wb'))

        #subiendo el archivo al campo longblob
        self._cn.UpdateBlobFile('proyeccion','vector',archivoTmp,'id=%d' % id)

        self._cn.Close()

    def ProyeccionRecuperar(self,id):
        self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)

        vector = self._cn.GetOne('select vector from proyeccion where id=%d' % id)
        archivoTmp = os.path.join(Configuracion.dirTmp,'proyeccion.txt')
        f = open(archivoTmp,'wb')
        f.write(vector)
        f.close()

        self._cn.Close()
        v = pickle.load(open(archivoTmp))
        return v

    def ProyeccionLimpiar(self):
        self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)
        sql = 'truncate table proyeccion'
        self._cn.Execute(sql)
        self._cn.Close()

    def ProyeccionGuardarMatriz(self,matriz):
        self.ProyeccionLimpiar()
        matriz = numpy.transpose(matriz)
        x,y = matriz.shape
        for i in range(x):
            vector = matriz[i]
            self.ProyeccionGuardar(vector)

    def ProyeccionRecuperarMatriz(self):
        matriz = None
        cr = self.ProyeccionId()
        for i in cr:
```

```
            v = self.ProyeccionRecuperar(i)
            v = v.reshape((v.size,1))
            if matriz==None:
                matriz = v
            else:
                matriz = numpy.concatenate((matriz,v),1)
        return matriz


class Rostro:
    def __init__(self):
        self._cn = adodb.NewADOConnection('mysql')

    def NoRostro(self,idPersona):
        self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)
        sql = 'select count(*) from rostro where idPersona=%d' % (idPersona)
        cur = self._cn.Execute(sql)
        resultado = 0
        while not cur.EOF:
            row = cur.fields
            resultado = row[0]
            cur.MoveNext()
        cur.Close()
        self._cn.Close()
        return resultado

    def MaxSize(self):
        size = (1,1)
        self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)
        sql = 'select max(ysize), max(xsize) from rostro'
        cur = self._cn.Execute(sql)
        while not cur.EOF:
            row = cur.fields
            size = (row[0],row[1])
            cur.MoveNext()
        cur.Close()
        self._cn.Close()
        return size

    def Lista(self):
        self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)
        resultado = {}
        sql = 'select id, archivo from rostro'
        cur = self._cn.Execute(sql)
```

```
        while not cur.EOF:
            row = cur.fields
            resultado[row[0]]=row[1]
            cur.MoveNext()
        cur.Close()
        self._cn.Close()
        return resultado

    def ArchivosRostros(self,idPersona):
        resultado = {}
        self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)
        sql = 'select id,archivo from rostro where idPersona=%d' % (idPersona)
        cur = self._cn.Execute(sql)
        i = 0
        while not cur.EOF:
            row = cur.fields
            resultado[i] = (row[0],row[1])
            i += 1
            cur.MoveNext()
        cur.Close()
        self._cn.Close()
        return resultado

class Persona:
    def __init__(self):
        self._cn = adodb.NewADOConnection('mysql')

    def Consulta(self,id):
        self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)

        sql = 'select id, nombre1, nombre2, apellido1, apellido2 from persona where id=%d' % id
        cur = self._cn.Execute(sql)
        while not cur.EOF:
            row = cur.fields
            break

        cur.Close()
        self._cn.Close()
        return row

    def Detalle(self):
        rostro = Rostro()
        self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)
        resultado = []
```

```
        sql = 'select id, nombre1, nombre2, apellido1, apellido2 from persona'
        cur = self._cn.Execute(sql)
        while not cur.EOF:
            row = cur.fields
            cur.MoveNext()
            norostro = rostro.NoRostro(row[0])
            resultado.append([row[0],row[1],row[2],row[3],row[4],norostro,False])

        cur.Close()
        self._cn.Close()
        return resultado

    def Resumen(self):
        self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)
        resultado = []
        sql = 'select id, nombre1, nombre2, apellido1, apellido2, count(rid) as nfot \
            from ( \
            select persona.id, apellido1, apellido2, nombre1, nombre2, rostro.id as rid \
            from persona left join rostro \
            on persona.id = rostro.idPersona) a \
            group by id, apellido1, apellido2, nombre1, nombre2'
        cur = self._cn.Execute(sql)
        while not cur.EOF:
            r = cur.fields
            cur.MoveNext()
            resultado.append([r[0],r[1],r[2],r[3],r[4],r[5]])

        cur.Close()
        self._cn.Close()
        return resultado

    def Insertar(self,id,nombre1,nombre2,apellido1,apellido2):
        self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)
        if id!=0:
            sql = "update persona "
            sql += "set nombre1='%s', nombre2='%s', " % (nombre1,nombre2)
            sql += "apellido1='%s', apellido2='%s' " % (apellido1,apellido2)
            sql += "where id=%d " % (id)
            cur = self._cn.Execute(sql)
        else:
            sql = "insert into persona(nombre1,nombre2,apellido1,apellido2) "
            sql += "values('%s','%s','%s','%s')" % (nombre1,nombre2,apellido1,apellido2)
            cur = self._cn.Execute(sql)
            id = cur.Insert_ID()
```

```
        cur.Close()
        self._cn.Close()
        return id

    def Eliminar(self,id):
        self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)
        sql = 'delete from persona where id=%d' % (id)
        cur = self._cn.Execute(sql)
        cur.Close()
        self._cn.Close()

    def AgregarFotografia(self,idpersona,fotografia):
        self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)
        sql = "insert into rostro(idPersona,archivo) values(%d,'%s')" % (idpersona,fotografia)
        cur = self._cn.Execute(sql)
        cur.Close()
        self._cn.Close()

    def ModificarFotografia(self,idRostro,fotografia,size):
        self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)
        sql = "update rostro set archivo='%s',xsize=%d,ysize=%d where id=%d" % (fotografia,size[0],size[1],idRostro)
        cur = self._cn.Execute(sql)
        cur.Close()
        self._cn.Close()

    def EliminarFotografia(self,idPersona,rostro):
        self._cn.Connect(Configuracion.HOST,Configuracion.USER,Configuracion.PASSWD,Configuracion.DB)
        sql = "delete from rostro where idPersona=%d and archivo='%s'" % (idPersona,rostro)
        cur = self._cn.Execute(sql)
        cur.Close()
        self._cn.Close()
```

## 17.1.6. Archivo: ide/Entrenar.py

```
# libreria WxPython
import numpy
import wx
import wx.grid as gridlib
# libreria PCA
import PCA
# libreria os
```

```
import os
# interfaz con BD
import BD
import EntrenarGrid
# utilerias que combinan wxPython con PIL
import imgutil
# recursos de la aplicacion
import images
import Configuracion
import datetime

persona = BD.Persona()
rostro = BD.Rostro()
rostroProcesado = BD.RostroProcesado()

# creando un objeto vector promedio
vp  = PCA.VectorPromedio()
# creando un objeto transformada KLT
klt = PCA.KLT()
# creando objeto para guardar vectores en BD
vector = BD.Vectores()

class Panel(wx.Panel):
   def __init__(self,parent):
      wx.Panel.__init__(self,parent,-1)

      # crea el grid, le pasa como parametros la interfaz de la base de datos Persona
      self._pGrid = wx.Window(self,-1,size=(550,300),style=wx.SIMPLE_BORDER)
      self._pGrid.SetBackgroundColour('STEEL BLUE')
      self._pGrid.SetPosition((10,10))
      self._grid = EntrenarGrid.GridPersona(self._pGrid)
      self._grid.SetSize((540,290))
      self._grid.SetPosition((5,5))

      # crea el grid de actividades
      self._pAct = wx.Window(self,-1,size=(510,200),style=wx.SIMPLE_BORDER)
      self._pAct.SetBackgroundColour('STEEL BLUE')
      self._pAct.SetPosition((50,315))
      self._gAct = lEntrenarGrid.GridActividades( self._pAct)
      self._gAct.SetSize((500,190))
      self._gAct.SetPosition((5,5))

      bEntrenar = wx.BitmapButton(self,-1,images.getmodificarBitmap(),(10,330),(35,35))
      bEntrenar.SetToolTipString('Entrenar modulo PCA')
```

```
        self.Bind(wx.EVT_BUTTON,self.OnEntrenar,bEntrenar)

    def OnEntrenar(self,evt):
        # encontrando la imagen promedio
        itemsProcesados = 0
        tiempo = datetime.datetime.now()
        self._gAct.GetTable().data[0][2] = tiempo.strftime('%Y/%m/%d-%H:%M:%S')
        self._pAct.Refresh()
        archivos = rostro.Lista()
        for a in archivos.keys():
            archivo = os.path.join(Configuracion.dirFot,archivos[a])
            vp.Promediar(PCA.ArchivoImagenAVector(archivo))
            itemsProcesados += 1
        tiempo = datetime.datetime.now()
        self._gAct.GetTable().data[0][3] = tiempo.strftime('%Y/%m/%d-%H:%M:%S')
        self._gAct.Refresh()

        # guardando imagen promedio
        imgprom = os.path.join(Configuracion.dirTmp,'imagenpromedio.png')
        PCA.VectorAArchivoImagen(imgprom,vp.Promedio())
        # guardando en BD
        vector.VPromedioGuardar(vp.Promedio())

        self._gAct.GetTable().data[0][4] = True
        self._gAct.GetTable().data[0][5] = 'Archivos procesados %d' % itemsProcesados
        self._gAct.Refresh()

        # econtrando las diferencias entre la imagen original y el promedio

        itemsProcesados = 0
        tiempo = datetime.datetime.now()
        self._gAct.GetTable().data[1][2] = tiempo.strftime('%Y/%m/%d-%H:%M:%S')
        self._gAct.Refresh()

        i = 0
        rostroProcesado.Limpiar()
        for a in archivos.keys():
            rostroProcesado.Registrar(i,a)
            i += 1
            archivo = os.path.join(Configuracion.dirFot,archivos[a])
            vec_dif = vp.Diferencia(PCA.ArchivoImagenAVector(archivo))
            # concatenando vectores para la transformada
            # Karhunen-Loeve
            klt.ConcatenarVector(vec_dif,a)
```

```
        itemsProcesados += 1
        # guarda diferencia
        #archivo_dif  = "./tmp/" + a
        #PCA.VectorAArchivoImagen(archivo_dif,vec_dif)

    # guardando la matriz de covarianza ampliada en la base de datos
    vector.CovarianzaAmpliadaGuardarMatriz(klt._A)

    tiempo = datetime.datetime.now()
    self._gAct.GetTable().data[1][3] = tiempo.strftime('%Y/%m/%d-%H:%M:%S')
    self._gAct.GetTable().data[1][4] = True
    self._gAct.GetTable().data[1][5] = 'Archivos procesados %d, Matriz generada: %d,%d (%d elementos)' \
       % (itemsProcesados, klt._A.shape[0],klt._A.shape[1],klt._A.shape[0]*klt._A.shape[1])
    self._gAct.Refresh()


    # rutinas para encontrar eigenvalores y eigenvectores
    tiempo = datetime.datetime.now()
    self._gAct.GetTable().data[2][2] = tiempo.strftime('%Y/%m/%d-%H:%M:%S')
    self._gAct.Refresh()
    #**********************************************************************
    klt.CalcularCovarianza()    # calculando covarianza
    #**********************************************************************

    # guardando la matriz de covarianza reducida en la base de datos
    vector.CovarianzaReducidaGuardarMatriz(klt._C)
    v = vector.CovarianzaReducidaRecuperarMatriz()

    tiempo = datetime.datetime.now()
    self._gAct.GetTable().data[2][3] = tiempo.strftime('%Y/%m/%d-%H:%M:%S')
    self._gAct.GetTable().data[2][4] = True
    self._gAct.GetTable().data[2][5] = 'Dimensiones: %d,%d (%d elementos)' \
       % (klt._C.shape[0],klt._C.shape[1],klt._C.shape[0]*klt._C.shape[1])
    self._gAct.Refresh()

    tiempo = datetime.datetime.now()
    self._gAct.GetTable().data[3][2] = tiempo.strftime('%Y/%m/%d-%H:%M:%S')
    self._gAct.Refresh()
    #**********************************************************************
    klt.CalcularEigenRed()     # calculando eigen de covarianz reducida
    #**********************************************************************
    tiempo = datetime.datetime.now()
    self._gAct.GetTable().data[3][3] = tiempo.strftime('%Y/%m/%d-%H:%M:%S')
    self._gAct.GetTable().data[3][4] = True
```

```
self._gAct.GetTable().data[3][5] = 'Dimensiones: %d,%d (%d elementos)' \
   % (klt._EVecRed.shape[0],klt._EVecRed.shape[1],klt._EVecRed.shape[0]*klt._EVecRed.shape[1])
self._gAct.Refresh()

tiempo = datetime.datetime.now()
self._gAct.GetTable().data[4][2] = tiempo.strftime('%Y/%m/%d-%H:%M:%S')
self._gAct.Refresh()
#*********************************************************************
klt.CalcularEigenvalores() # calculando eigenvalores
#*********************************************************************
# guardando eigenvalores en la BD
vector.EigenvalorGuardar(klt._eigenvalor)

tiempo = datetime.datetime.now()
self._gAct.GetTable().data[4][3] = tiempo.strftime('%Y/%m/%d-%H:%M:%S')
self._gAct.GetTable().data[4][4] = True
self._gAct.GetTable().data[4][5] = 'Dimensiones: %d,%d (%d elementos)' \
   % (1,klt._eigenvalor.shape[0],klt._eigenvalor.size)


tiempo = datetime.datetime.now()
self._gAct.GetTable().data[5][2] = tiempo.strftime('%Y/%m/%d-%H:%M:%S')
self._gAct.Refresh()
#*********************************************************************
klt.CalcularEigenvectores() # calculando eigenvectores
#*********************************************************************
# guardando eigenvectores en la BD
vector.EigenvectorGuardarMatriz(klt._eigenvector)

tiempo = datetime.datetime.now()
self._gAct.GetTable().data[5][3] = tiempo.strftime('%Y/%m/%d-%H:%M:%S')
self._gAct.GetTable().data[5][4] = True
self._gAct.GetTable().data[5][5] = 'Dimensiones: %d,%d (%d elementos)' \
   % (klt._eigenvector.shape[0],klt._eigenvector.shape[1],klt._eigenvector.shape[0]*klt._eigenvector.shape[1])

tiempo = datetime.datetime.now()
self._gAct.GetTable().data[6][2] = tiempo.strftime('%Y/%m/%d-%H:%M:%S')
self._gAct.Refresh()
#*********************************************************************
klt.CalcularProyecciones()
#*********************************************************************

# guardar proyecciones en BD
vector.ProyeccionGuardarMatriz(klt._proyeccion)
```

```
        tiempo = datetime.datetime.now()
        self._gAct.GetTable().data[6][3] = tiempo.strftime('%Y/%m/%d-%H:%M:%S')
        self._gAct.GetTable().data[6][4] = True
        self._gAct.GetTable().data[6][5] = 'Dimensiones: %d,%d (%d elementos)' \
           % (klt._proyeccion.shape[0],klt._proyeccion.shape[1],klt._proyeccion.shape[0]*klt._proyeccion.shape[1])

        # rutinas para guardar los eigenfaces mas representativos
        # (con mayor valor de eigenvalues)
        j = 0
        for i in klt.SecuenciaEigenvaluesRepr(klt._n):
           eigenface = klt.Eigenface(i)
           archivo_eig = 'eigenface_%d.png' % j
           archivo_eig = os.path.join(Configuracion.dirTmp,archivo_eig)
           PCA.VectorAArchivoImagen(archivo_eig,eigenface)
           j += 1

        # reconstruyendo la imagen 10 utilizando 15 eigenvectores repr.
        #PCA.VectorAArchivoImagen("./tmp/reconstruccion.jpg",
           #self._klt.ReconstruirImagen(self._vp.Promedio(),10,15))

        # ***** identificacion de una imagen
        # calculando la proyeccion de la imagen reconstruida
        #r=PCA.ArchivoImagenAVector("./tmp/reconstruccion.jpg")
        #self._klt.CalcularProyeccionesImagen(r,self._vp.Promedio())

        # probando metodo de la distancia euclidea
        #self._klt.CalcularDistanciaEuclidea()
        #print "self._klt._euclidea: ", self._klt._euclidea
        #print "identificacion: ", self._klt.IdentificacionDistanciaEuclidea()

        #x = range(1,self._klt._euclidea.size+1)
        #grafico = pylab.plot(x,self._klt._euclidea)
        #pylab.show()


class Frame(wx.MDIChildFrame):
   def __init__(
      self,parent,ID,title,pos=wx.DefaultPosition,size=wx.DefaultSize,
      style=wx.DEFAULT_FRAME_STYLE):

      wx.MDIChildFrame.__init__(self,parent,ID,title,pos,size,style)
      self._icon = images.getpythonIcon()
      self.SetIcon(self._icon)
```

```
        self._panel = Panel(self)
        self.Bind(wx.EVT_CLOSE,self.OnCloseWindow)

    def OnCloseWindow(self,evt):
        self.Destroy()
```

## 17.1.7. Archivo: ide/EntrenarGrid.py

```
import wx.grid as gridlib

import BD

persona = BD.Persona()

class GridPersonaE(gridlib.PyGridTableBase):
    def __init__(self,parent):
        gridlib.PyGridTableBase.__init__(self)
        self._colslabels = ['ID','Primer Nombre','Segundo Nombre',
            'Primer Apellido','Segundo Apellido','No Fotos']
        self._datatypes = [
            gridlib.GRID_VALUE_NUMBER,
            gridlib.GRID_VALUE_STRING,
            gridlib.GRID_VALUE_STRING,
            gridlib.GRID_VALUE_STRING,
            gridlib.GRID_VALUE_STRING,
            gridlib.GRID_VALUE_NUMBER]

        self._data = persona.Resumen()
        self.odd = gridlib.GridCellAttr()
        self.odd.SetBackgroundColour('WHITE')
        self.even = gridlib.GridCellAttr()
        self.even.SetBackgroundColour('KHAKI')

    def GetAttr(self,row,col,kind):
        attr = [self.even,self.odd][row%2]
        attr.IncRef()
        return attr

    def GetNumberRows(self):
        return len(self._data)

    def GetNumberCols(self):
        return len(self._colslabels)
```

```
    def IsEmptyCell(self,row,col):
        try:
            return not self._data[row][col]
        except IndexError:
            return True

    def GetValue(self,row,col):
        try:
            return self._data[row][col]
        except IndexError:
            return ''

    def GetColLabelValue(self,col):
        return self._colslabels[col]

    def GetTypeName(self,row,col):
        return self._datatypes[col]

    def CanGetValueAs(self,row,col,typeName):
        colType = self._datatypes[col].split(':')[0]
        if typeName == colType:
            return True
        else:
            return False

    def CanSetValueAs(self,row,col,typeName):
        return self.CanGetValueAs(row,col,typeName)

class GridPersona(gridlib.Grid):
    def __init__(self,parent):
        gridlib.Grid.__init__(self,parent,-1)
        self._gridE = GridPersonaE(self)
        self.SetTable(self._gridE,True)

        self.SetRowLabelSize(0)
        self.SetMargins(0,0)
        self.AutoSizeColumns(False)


class GridActividadesE(gridlib.PyGridTableBase):
    def __init__(self):
        gridlib.PyGridTableBase.__init__(self)
        self.colLabels = ['ID','Actividad','Inicio','Fin','Completada','Resumen']
```

```
        self.dataTypes = [gridlib.GRID_VALUE_NUMBER,
                          gridlib.GRID_VALUE_STRING,
                          gridlib.GRID_VALUE_STRING,
                          gridlib.GRID_VALUE_STRING,
                          gridlib.GRID_VALUE_BOOL,
                          gridlib.GRID_VALUE_STRING]

        self.data = [
                     [1,'Calculo de vector promedio','','',0,''],
                     [2,'Calculo diferencias entre img. original y vector promedio','','',0,''],
                     [3,'Calculo de matriz de Covarianza','','',0,''],
                     [4,'Calculo de matriz de Eigenvectores en matriz de covarianza (Eigenvectores Reducidos)','','',0,''],
                     [5,'Calculo de Eigenvalores a partir de Eigenvectores de covarianza','','',0,''],
                     [6,'Calculo de Eigenvectores a partir de Eigenvalores y matriz reducida','','',0,''],
                     [7,'Calculo de Proyecciones sobre Eigenfaces','','',0,'']]

    def GetNumberRows(self):
        return len(self.data)

    def GetNumberCols(self):
        return len(self.data[0])

    def IsEmptyCell(self,row,col):
        try:
            return not self.data[row][col]
        except IndexError:
            return True

    def GetValue(self,row,col):
        try:
            return self.data[row][col]
        except IndexError:
            return ''

    def GetColLabelValue(self,col):
        return self.colLabels[col]

    def GetTypeName(self,row,col):
        return self.dataTypes[col]

    def CanGetValueAs(self,row,col,typeName):
        colType = self.dataTypes[col].split(':')[0]
        if typeName == colType:
            return True
```

```
        else:
            return False

    def CanSetValueAs(self,row,col,typeName):
        return self.CanGetValueAs(row,col,typeName)


class GridActividades(gridlib.Grid):
    def __init__(self,parent):
        gridlib.Grid.__init__(self,parent)
        table = GridActividadesE()
        self.SetTable(table,True)
        self.SetRowLabelSize(0)
        self.SetMargins(0,0)
        self.AutoSizeColumns(True)
```

## 17.1.8. Archivo: ide/Grid.py

```
import wx
import wx.grid as gridlib

# ***************************************************************************
# Las clases: Table, MiGrid, FrameGrid son empleadas para visualizar
# de forma grafica el contenido de cualquier vector; se utilizan especialmente
# para observar la matriz de covarianza, de proyecciones, de eigenvalores
# y eigenvectores

class Table(gridlib.PyGridTableBase):
    def __init__(self,vector):
        gridlib.PyGridTableBase.__init__(self)
        self._vector = vector
    def GetNumberRows(self):
        r,c = self._vector.shape
        return r
    def GetNumberCols(self):
        r,c = self._vector.shape
        return c
    def IsEmptyCell(self, row, col):
        return False
    def GetValue(self, row, col):
        return '%(valor)3.6f' % {'valor': self._vector[row][col]}
    def SetValue(self, row, col, value):
```

```
        pass
    def GetColLabelValue(self, col):
        return str(col+1)

class MiGrid(gridlib.Grid):
    def __init__(self,parent,vector):
        gridlib.Grid.__init__(self,parent,-1)
        table = Table(vector)
        self.SetTable(table, True)

class Frame(wx.MDIChildFrame):
    def __init__(
        self,parent,ID,title,vector,pos=wx.DefaultPosition,size=wx.DefaultSize,
        style=wx.DEFAULT_FRAME_STYLE):
        wx.MDIChildFrame.__init__(self,parent,ID,title,pos,size,style)
        panel = wx.Panel(self,-1)
        self.grid = MiGrid(self,vector)
        self.grid.SetSize((size[0]-10,size[1]-35))
        self.Bind(wx.EVT_CLOSE,self.OnCloseWindow)
        self.Bind(wx.EVT_SIZE,self.OnSizeWindow)

    def OnCloseWindow(self,evt):
        self.Destroy()

    def OnSizeWindow(self,evt):
        size = evt.GetSize()
        self.SetSize(size)
        self.grid.SetSize((size[0]-10,size[1]-35))
```

## 17.1.9. Archivo: ide/Identificar.py

```
import PCA
import BD
import wx
import images
import numpy
import Configuracion
import os
import imgutil
import sys
import string
```

```
vector = BD.Vectores()
klt = PCA.KLT()

class Frame(wx.MDIChildFrame):
    def __init__(
        self,parent,ID,title,pos=wx.DefaultPosition,size=wx.DefaultSize,
        style=wx.DEFAULT_FRAME_STYLE):

        wx.MDIChildFrame.__init__(self,parent,ID,title,pos,size,style)
        self._icon = images.getpythonIcon()
        self.SetIcon(self._icon)

        self._panel = Panel(self)
        self.Bind(wx.EVT_CLOSE,self.OnCloseWindow)

    def OnCloseWindow(self,evt):
        self.Destroy()

class PanelImagen(wx.Window):
    def __init__(self,parent):
        wx.Window.__init__(self,parent,-1,style=wx.SIMPLE_BORDER)

        r = images.getfondoBitmap()
        x,y = r.GetWidth(),r.GetHeight()

        self._fondo = wx.StaticBitmap(self,-1,r,(0,0),(x,y))
        self._imagen = wx.StaticBitmap(self,-1,r,(0,0),(x,y))

    def Mostrar(self,rostro):
        if rostro!=None:
            try:
                r = imgutil.GetBitmap(rostro)
                x,y = r.GetWidth(),r.GetHeight()
                self._imagen = wx.StaticBitmap(self,-1,r,(0,0),(x,y))
                self._imagen.Centre()
                self._imagen.Show(True)
            except:
                self._imagen.Show(False)
        else:
            r = images.getfondoBitmap()
            x,y = r.GetWidth(),r.GetHeight()
            self._imagen = wx.StaticBitmap(self,-1,r,(0,0),(x,y))
        self.Refresh()
```

```
class PanelLista(wx.Window):
    def __init__(self,parent):
        wx.Window.__init__(self,parent,-1,style=wx.SIMPLE_BORDER)
        w,h = self.GetClientSizeTuple()

        self._lista = wx.ListCtrl(self,-1,(0,0),(w,h),
                        style=wx.LC_REPORT)
        self._lista.InsertColumn(0,'Id Foto')
        self._lista.InsertColumn(1,'Dif. Euclidea')
        self._lista.InsertColumn(2,'%Certeza')

        self.Bind(wx.EVT_SIZE, self.OnSize)
        self.Bind(wx.EVT_LIST_ITEM_SELECTED, self.OnListaSelected, self._lista)

        self._pReconstruir = None # la lista debe saber a donde reconstruir la imagen
        self._pReconstruirTxt = None # control de texto con la identificacion

    def OnSize(self, evt):
        w,h = self.GetClientSizeTuple()
        self._lista.SetDimensions(0, 0, w, h)

    def OnListaSelected(self,evt):
        currentItem = evt.m_itemIndex
        idFoto = string.atof(self._lista.GetItem(currentItem,0).GetText())
        archivo = os.path.join(Configuracion.dirTmp,'reconstruccion.png')
        vectorPromedio = vector.VPromedioRecuperar()

        imagenReconstruida = klt.ReconstruirImagen(
                                vectorPromedio,
                                idFoto,
                                klt._n_eigenvector)
        PCA.VectorAArchivoImagen(archivo,imagenReconstruida)
        self._pReconstruir.Mostrar(archivo)

        rostroProcesado = BD.RostroProcesado()
        mensaje = '%d %s %s %s %s' % (
            rostroProcesado.Persona(idFoto))
        self._pReconstruirTxt.SetValue(mensaje)

class Panel(wx.Panel):
    def __init__(self,parent):
        wx.Panel.__init__(self,parent,-1)

        self._pImagen = PanelImagen(self)
```

```
        self._pImagen.SetPosition((10,10))
        self._pImagen.SetSize((300,300))

        bSeleccionarImagen = wx.BitmapButton(self,-1,images.getexploreBitmap(),
                                (10,315),(35,35))
        bSeleccionarImagen.SetToolTipString('Seleccione rostro a buscar')
        wx.StaticText(self,-1,'Seleccione una imagen existente para la busqueda',
           (50,325))

        self.Bind(wx.EVT_BUTTON,self.OnSeleccionar,bSeleccionarImagen)

        self._pImagenR = PanelImagen(self)
        self._pImagenR.SetPosition((570,10))
        self._pImagenR.SetSize((300,300))

        self._persona = wx.TextCtrl(self, -1, '',
                        size=(295, -1),pos=(570,325))

        self._pLista = PanelLista(self)
        self._pLista.SetPosition((315,10))
        self._pLista.SetSize((250,300))
        self._pLista._pReconstruir = self._pImagenR  #para reconstruir la imagen a partir de evento
        self._pLista._pReconstruirTxt = self._persona # para identificar la reconstruccion

        bBuscar = wx.BitmapButton(self,-1,images.getbuscar2Bitmap(),
                        (315,315),(35,35))


        self.Bind(wx.EVT_BUTTON,self.OnBuscar,bBuscar)

        # reconstruyendo objetos necesarios para el reconocimiento facial
        # creando el objeto KLT reconstruyendolo de la base de datos
        self._vectorPromedio = vector.VPromedioRecuperar()
        klt._eigenvector = vector.EigenvectorRecuperarMatriz()
        klt._n_eigenvector = klt._eigenvector.shape[1]
        klt._eigenvalor = vector.EigenvalorRecuperar()
        klt._proyeccion = vector.ProyeccionRecuperarMatriz()
        klt._n = klt._eigenvalor.size

        self._imagenDesconocida = ''

    def OnSeleccionar(self,evt):
        dlg = wx.FileDialog(
                self,message='Seleccione la imagen',defaultDir=os.getcwd(),
```

```
            defaultFile='',wildcard=Configuracion.wildcard,style=wx.OPEN|wx.MULTIPLE|wx.CHANGE_DIR)
        if dlg.ShowModal()==wx.ID_OK:
            paths = dlg.GetPaths()
            for path in paths:
                self._pImagen.Mostrar(path)
                self._imagenDesconocida = path
        dlg.Destroy()

    def OnBuscar(self,evt):
        vectorDesconocido = PCA.ArchivoImagenAVector(self._imagenDesconocida)
        klt.CalcularProyeccionesImagen(vectorDesconocido,self._vectorPromedio)
        klt.CalcularDistanciaEuclidea()
        self._matrizDistancia = klt.MatrizDistancia()
        self.ActualizarLista()

    def ActualizarLista(self):
        self._pLista._lista.DeleteAllItems()
        f,c = self._matrizDistancia.shape
        for i in range(f):
            idx = self._pLista._lista.InsertStringItem(sys.maxint,str(self._matrizDistancia[i][0]))
            self._pLista._lista.SetStringItem(idx,1,str(self._matrizDistancia[i][1]))
            self._pLista._lista.SetStringItem(idx,2,str(self._matrizDistancia[i][2]))
```

## 17.1.10. Archivo: ide/ImagenPromedio.py

```
import wx
import images
import PCA
import BD
import os
import Configuracion
import imgutil

vector = BD.Vectores()

class Frame(wx.MDIChildFrame):
    def __init__(
        self,parent,ID,title,pos=wx.DefaultPosition,size=wx.DefaultSize,
        style=wx.DEFAULT_FRAME_STYLE):

        wx.MDIChildFrame.__init__(self,parent,ID,title,pos,size,style)
```

```
        self._icon = images.getpythonIcon()
        self.SetIcon(self._icon)

        self._panel = Panel(self)
        self.Bind(wx.EVT_CLOSE,self.OnCloseWindow)

    def OnCloseWindow(self,evt):
        self.Destroy()

class Panel(wx.Panel):
    def __init__(self,parent):
        wx.Panel.__init__(self,parent,-1)
        self._pImagen = PanelImagen(self)
        self._pImagen.SetPosition((10,10))
        self._pImagen.SetSize((320,350))

        bActualizar = wx.BitmapButton(self,-1,images.getactualizarBitmap(),(10,370))
        bActualizar.SetToolTipString('Visualizar imagen promedio')
        self.Bind(wx.EVT_BUTTON,self.OnActualizar,bActualizar)

        wx.StaticText(self,-1,'Visualizacion de Imagen promedio calculada',(60,375))

    def OnActualizar(self,evt):
        archivoTmp = 'imagenpromedio.png'
        archivoTmp = os.path.join(Configuracion.dirTmp,archivoTmp)

        v = vector.VPromedioRecuperar()
        PCA.VectorAArchivoImagen(archivoTmp,v)

        self._pImagen.Mostrar(archivoTmp)
        self._pImagen.Refresh()


class PanelImagen(wx.Window):
    def __init__(self,parent):
        wx.Window.__init__(self,parent,-1,style=wx.SIMPLE_BORDER)

        r = images.getfondoBitmap()
        x,y = r.GetWidth(),r.GetHeight()

        self._fondo = wx.StaticBitmap(self,-1,r,(0,0),(x,y))
        self._imagen = wx.StaticBitmap(self,-1,r,(0,0),(x,y))

    def Mostrar(self,imagen):
```

```
        if imagen!=None:
            try:
                r = imgutil.GetBitmap(imagen)
                x,y = (r.GetWidth(),r.GetHeight())
                self._imagen = wx.StaticBitmap(self,-1,r,(0,0),(x,y))
                self._imagen.Centre()
                self._imagen.Show(True)
            except:
                self._imagen.Show(False)
        else:
            r = images.getfondoBitmap()
            x,y = r.GetWidth(),r.GetHeight()
            self._imagen = wx.StaticBitmap(self,-1,r,(0,0),(x,y))
        self.Refresh()
```

## 17.1.11. Archivo: ide/images.py

```
#----------------------------------------------------------------------
# This file was generated by E:\tesis\pyEigen2\ide\res\resources.py
#
from wx import ImageFromStream, BitmapFromImage
from wx import EmptyIcon
import cStringIO


def getGridBGData():
    return \
'BINARY DATA'

def getGridBGBitmap():
    return BitmapFromImage(getGridBGImage())

def getGridBGImage():
    stream = cStringIO.StringIO(getGridBGData())
    return ImageFromStream(stream)

def getGridBGIcon():
    icon = EmptyIcon()
    icon.CopyFromBitmap(getGridBGBitmap())
    return icon

#----------------------------------------------------------------------
```

```
def getactualizarData():
    return \
'BINARY DATA'

def getactualizarBitmap():
    return BitmapFromImage(getactualizarImage())

def getactualizarImage():
    stream = cStringIO.StringIO(getactualizarData())
    return ImageFromStream(stream)

def getactualizarIcon():
    icon = EmptyIcon()
    icon.CopyFromBitmap(getactualizarBitmap())
    return icon

#----------------------------------------------------------------------
def getagregarData():
    return \
'BINARY DATA'

def getagregarBitmap():
    return BitmapFromImage(getagregarImage())

def getagregarImage():
    stream = cStringIO.StringIO(getagregarData())
    return ImageFromStream(stream)

def getagregarIcon():
    icon = EmptyIcon()
    icon.CopyFromBitmap(getagregarBitmap())
    return icon

#----------------------------------------------------------------------
def getanteriorData():
    return \
'BINARY DATA'

def getanteriorBitmap():
    return BitmapFromImage(getanteriorImage())

def getanteriorImage():
    stream = cStringIO.StringIO(getanteriorData())
    return ImageFromStream(stream)
```

```
def getanteriorIcon():
    icon = EmptyIcon()
    icon.CopyFromBitmap(getanteriorBitmap())
    return icon

#----------------------------------------------------------------------
def getborrarData():
    return \
'BINARY DATA'

def getborrarBitmap():
    return BitmapFromImage(getborrarImage())

def getborrarImage():
    stream = cStringIO.StringIO(getborrarData())
    return ImageFromStream(stream)

def getborrarIcon():
    icon = EmptyIcon()
    icon.CopyFromBitmap(getborrarBitmap())
    return icon

#----------------------------------------------------------------------
def getbuscarData():
    return \
'BINARY DATA'

def getbuscarBitmap():
    return BitmapFromImage(getbuscarImage())

def getbuscarImage():
    stream = cStringIO.StringIO(getbuscarData())
    return ImageFromStream(stream)

def getbuscarIcon():
    icon = EmptyIcon()
    icon.CopyFromBitmap(getbuscarBitmap())
    return icon

#----------------------------------------------------------------------
def getbuscar2Data():
    return \
'BINARY DATA'
```

```
def getbuscar2Bitmap():
    return BitmapFromImage(getbuscar2Image())

def getbuscar2Image():
    stream = cStringIO.StringIO(getbuscar2Data())
    return ImageFromStream(stream)

def getbuscar2Icon():
    icon = EmptyIcon()
    icon.CopyFromBitmap(getbuscar2Bitmap())
    return icon

#----------------------------------------------------------------------
def getexploreData():
    return \
'BINARY DATA'

def getexploreBitmap():
    return BitmapFromImage(getexploreImage())

def getexploreImage():
    stream = cStringIO.StringIO(getexploreData())
    return ImageFromStream(stream)

def getexploreIcon():
    icon = EmptyIcon()
    icon.CopyFromBitmap(getexploreBitmap())
    return icon

#----------------------------------------------------------------------
def getfondoData():
    return \
'BINARY DATA'

def getfondoBitmap():
    return BitmapFromImage(getfondoImage())

def getfondoImage():
    stream = cStringIO.StringIO(getfondoData())
    return ImageFromStream(stream)

def getfondoIcon():
    icon = EmptyIcon()
```

```
    icon.CopyFromBitmap(getfondoBitmap())
    return icon

#----------------------------------------------------------------------
def getmodificarData():
    return \
'BINARY DATA'

def getmodificarBitmap():
    return BitmapFromImage(getmodificarImage())

def getmodificarImage():
    stream = cStringIO.StringIO(getmodificarData())
    return ImageFromStream(stream)

def getmodificarIcon():
    icon = EmptyIcon()
    icon.CopyFromBitmap(getmodificarBitmap())
    return icon

#----------------------------------------------------------------------
def getpythonData():
    return \
'BINARY DATA'

def getpythonBitmap():
    return BitmapFromImage(getpythonImage())

def getpythonImage():
    stream = cStringIO.StringIO(getpythonData())
    return ImageFromStream(stream)

def getpythonIcon():
    icon = EmptyIcon()
    icon.CopyFromBitmap(getpythonBitmap())
    return icon

#----------------------------------------------------------------------
def getsiguienteData():
    return \
'BINARY DATA'

def getsiguienteBitmap():
    return BitmapFromImage(getsiguienteImage())
```

```
def getsiguienteImage():
    stream = cStringIO.StringIO(getsiguienteData())
    return ImageFromStream(stream)

def getsiguienteIcon():
    icon = EmptyIcon()
    icon.CopyFromBitmap(getsiguienteBitmap())
    return icon
```

## 17.1.12. Archivo: ide/imgutil.py

```
import wx
import Image

#fuente: http://wiki.wxpython.org/index.cgi/WorkingWithImages

def bitmapToPil(bitmap):
    return imageToPil(bitmapToImage(bitmap))

def bitmapToImage(bitmap):
    return wx.wxImageFromBitmap(bitmap)

def pilToBitmap(pil):
    return imageToBitmap(pilToImage(pil))

def pilToImage(pil):
    image = wx.wxEmptyImage(pil.size[0], pil.size[1])
    image.SetData(pil.convert('RGB').tostring())
    return image

#Or, if you want to copy alpha cannels too (available from wxPython 2.5)
def piltoimage(pil, alpha=True):
    if alpha:
        image = apply( wx.EmptyImage, pil.size )
        image.SetData( pil.convert( 'RGB' ).tostring()[3::4])
    else:
        image = wx.EmptyImage( pil.size[0], pil.size[1] )
        new_image = pil.convert('RGB')
        data = new_image.tostring()
        image.SetData(data)
    return image
```

```
def imageToPil(image):
    pil = Image.new('RGB',(image.GetWidth(), image.GetHeight()))
    pil.fromstring(image.GetData())
    return pil

def imageToBitmap(image):
    return image.ConvertToBitmap()

def GetBitmap(archivo):
    source = Image.open( archivo, 'r' )
    image = wx.EmptyImage(source.size[0],source.size[1])
    image.SetData( source.convert( 'RGB' ).tostring())
    #if the image has an alpha channel, you can set it with this line:
    image.SetAlphaData( source.convert('RGBA').tostring()[3::4])
    return image.ConvertToBitmap()

def GetSize(archivo):
    source = Image.open( archivo, 'r')
    size = source.size
    return size
```

## 17.1.13. Archivo: ide/MntPersona.py

```
import wx
import imgutil
import images
import MntPersonaGrid
import BD
import sys
import Configuracion
import os
import datetime
import Image

persona = BD.Persona()
rostro = BD.Rostro()

class Frame(wx.MDIChildFrame):
    def __init__(self,parent,ID,title,pos=wx.DefaultPosition,
        size=wx.DefaultSize,style=wx.DEFAULT_FRAME_STYLE):
        wx.MDIChildFrame.__init__(self,parent,ID,title,pos,size,style)
        self._panel = Panel(self)
        self._icon = images.getpythonIcon()
```

```
        self.SetIcon(self._icon)


class Panel(wx.Panel):
    def __init__(self,parent):
        wx.Panel.__init__(self,parent,-1)
        self._init()

    def MostrarRostro(self,id):
        self._rostros = rostro.ArchivosRostros(id)
        self._i_rostros = 0
        if len(self._rostros)>0:
            foto = os.path.join(Configuracion.dirFot,self._rostros[0][1])
            self._rostroActual = self._rostros[0][1]
            self._pImagen.Mostrar(foto)
        else:
            self._rostroActual = ''
            self._pImagen.Mostrar(None)

    def _init(self):
        #self._pImagen = wx.Window(self,-1,size=(350,350),style=wx.SIMPLE_BORDER)
        #self._pImagen.SetPosition((10,10))
        #self._pImagen._sb = wx.StaticBitmap(self._pImagen,-1,images.getfondoBitmap(),(10,10))
        self._pImagen = PanelImagen(self)
        self._pImagen.SetPosition((10,10))
        self._pImagen.SetSize((350,350))

        self._pGrid = wx.Window(self,-1,size=(510,350),style=wx.SIMPLE_BORDER)
        self._pGrid.SetBackgroundColour('STEEL BLUE')
        self._pGrid.SetPosition((370,10))
        self._grid = MntPersonaGrid.GridPersona(self._pGrid,self.MostrarRostro)
        x,y = self._pGrid.GetSize()
        self._grid.SetSize((x-10,y-10))
        self._grid.SetPosition((5,5))

        banterior = wx.BitmapButton(self,-1,images.getanteriorBitmap(),(10,370),(35,35))
        banterior.SetToolTipString('Rostro anterior')
        bsiguiente = wx.BitmapButton(self,-1,images.getsiguienteBitmap(),(45,370),(35,35))
        bsiguiente.SetToolTipString('Rostro siguiente')
        bagregar = wx.BitmapButton(self,-1,images.getagregarBitmap(),(90,370),(35,35))
        bagregar.SetToolTipString('Agregar rostro')
        bactualizar = wx.BitmapButton(self,-1,images.getactualizarBitmap(),(125,370),(35,35))
        bactualizar.SetToolTipString('Actualizar rostro')
        beliminar = wx.BitmapButton(self,-1,images.getborrarBitmap(),(160,370),(35,35))
```

```
        beliminar.SetToolTipString('Eliminar rostro')

        self._rostros = [] #rostros asociados a la persona seleccionada
        self._i_rostros = 0 #id utilizado para el desplazamiento de rostros
        self._rostroActual = ''
        self.Bind(wx.EVT_BUTTON,self.OnRostroAnterior,banterior)
        self.Bind(wx.EVT_BUTTON,self.OnRostroSiguiente,bsiguiente)
        self.Bind(wx.EVT_BUTTON,self.OnRostroAgregar,bagregar)
        self.Bind(wx.EVT_BUTTON,self.OnRostroActualizar,bactualizar)
        self.Bind(wx.EVT_BUTTON,self.OnRostroEliminar,beliminar)

    def OnRostroAnterior(self,evt):
        if self._i_rostros>0:
            self._i_rostros -= 1
            foto = os.path.join(Configuracion.dirFot,self._rostros[self._i_rostros][1])
            self._rostroActual = self._rostros[self._i_rostros][1]
            self._pImagen.Mostrar(foto)

    def OnRostroSiguiente(self,evt):
        if self._i_rostros+1<len(self._rostros):
            self._i_rostros += 1
            foto = os.path.join(Configuracion.dirFot,self._rostros[self._i_rostros][1])
            self._rostroActual = self._rostros[self._i_rostros][1]
            self._pImagen.Mostrar(foto)

    def OnRostroAgregar(self,evt):
        dlg = wx.FileDialog(
                self,message='Seleccione la imagen',defaultDir=os.getcwd(),
                defaultFile='',wildcard=Configuracion.wildcard,style=wx.OPEN | wx.MULTIPLE | wx.CHANGE_DIR)
        if dlg.ShowModal()==wx.ID_OK:
            paths = dlg.GetPaths()
            for path in paths:
                archivo = os.path.basename(path)
                dt = datetime.datetime.now()
                na = '%s_%s.png' % (archivo,dt.strftime('%Y%m%d%H%M%S')) #nuevo archivo
                fa = os.path.join(Configuracion.dirFot,na)
                Image.open(path).save(fa)

                id = self._grid._idSeleccionado
                detalle = persona.Consulta(id)
                nombre1 = detalle[1]
                nombre2 = detalle[2]
                apellido1 = detalle[3]
                apellido2 = detalle[4]
```

```
            persona.AgregarFotografia(self._grid._idSeleccionado,na)

            msgdlg = wx.MessageDialog(self,'Rostro asociado a la persona: %d %s %s %s %s' %
              (id,nombre1,nombre2,apellido1,apellido2),'Rostro asociado',wx.OK | wx.ICON_INFORMATION)
            msgdlg.ShowModal()
            msgdlg.Destroy()
        dlg.Destroy()

    def OnRostroActualizar(self,evt):
        dlg = wx.FileDialog(
                self,message='Seleccione la imagen',defaultDir=os.getcwd(),
                defaultFile='',wildcard=Configuracion.wildcard,style=wx.OPEN|wx.MULTIPLE|wx.CHANGE_DIR)
        if dlg.ShowModal()==wx.ID_OK:
            paths = dlg.GetPaths()
            for path in paths:
                archivo = os.path.basename(path)
                dt = datetime.datetime.now()
                na = self._rostroActual # se sobreescribira un archivo en el directorio
                fa = os.path.join(Configuracion.dirFot,na)
                Image.open(path).save(fa)

                id = self._grid._idSeleccionado
                detalle = persona.Consulta(id)
                nombre1 = detalle[1]
                nombre2 = detalle[2]
                apellido1 = detalle[3]
                apellido2 = detalle[4]

                msgdlg = wx.MessageDialog(self,'Rostro actualizado a la persona: %d %s %s %s %s' %
                  (id,nombre1,nombre2,apellido1,apellido2),'Rostro actualizado',wx.OK | wx.ICON_INFORMATION)
                msgdlg.ShowModal()
                msgdlg.Destroy()
        dlg.Destroy()
        self._pImagen.Mostrar(fa)

    def OnRostroEliminar(self,evt):
        dlg = wx.MessageDialog(self,'Desea eliminar el rostro actual','Eliminar Rostro',
                wx.YES_NO | wx.ICON_INFORMATION)
        if dlg.ShowModal()==wx.ID_YES:
            persona.EliminarFotografia(self._grid._idSeleccionado,self._rostroActual)
            archivo = os.path.join(Configuracion.dirFot,self._rostroActual)
            os.remove(archivo)

            self.MostrarRostro(self._grid._idSeleccionado)
```

```
            msgdlg = wx.MessageDialog(self,'Eliminacion realizada','Eliminar Rostro',wx.OK | wx.ICON_INFORMATION)
            msgdlg.ShowModal()
            msgdlg.Destroy()
        else:
            msgdlg = wx.MessageDialog(self,'Eliminacion cancelada','Eliminar Rostro',wx.OK | wx.ICON_INFORMATION)
            msgdlg.ShowModal()
            msgdlg.Destroy()
        dlg.Destroy()

class PanelImagen(wx.Window):
    def __init__(self,parent):
        wx.Window.__init__(self,parent,-1,style=wx.SIMPLE_BORDER)

        r = images.getfondoBitmap()
        x,y = r.GetWidth(),r.GetHeight()

        self._fondo = wx.StaticBitmap(self,-1,r,(0,0),(x,y)) # utilizado para ocultar rostro
        self._imagen = wx.StaticBitmap(self,-1,r,(0,0),(x,y)) # utilizado para mostrar rostro actual

    def Mostrar(self,rostro):
        if rostro!=None:
            try:
                r = imgutil.GetBitmap(os.path.join(Configuracion.dirFot,rostro))
                x,y = (r.GetWidth(),r.GetHeight())
                self._imagen = wx.StaticBitmap(self,-1,r,(0,0),(x,y))
                self._imagen.Centre()
                self._imagen.Show(True)
            except:
                self._imagen.Show(False)
        else:
            r = images.getfondoBitmap()
            x,y = r.GetWidth(),r.GetHeight()
            self._imagen = wx.StaticBitmap(self,-1,r,(0,0),(x,y))
        self.Refresh()
```

## 17.1.14. Archivo: ide/MntPersonaGrid.py

```
import wx
import wx.grid as gridlib
import BD
import wx.lib.imagebrowser as ib
import os
import Image
```

```
import Configuracion

persona = BD.Persona()
rostro = BD.Rostro()

class GridPersona(gridlib.Grid):
    def __init__(self,parent,evento):
        gridlib.Grid.__init__(self,parent,-1)
        self._grid = GridPersonaE(self)
        self.SetTable(self._grid,True)
        self.SetRowLabelSize(0)
        self.SetMargins(0,0)
        self.AutoSizeColumns(False)

        self._idSeleccionado = 0
        self._rowSeleccionado = 0

        self._eventoSeleccionar = evento #evento que se dispara al cambiar de registro
        self.Bind(gridlib.EVT_GRID_SELECT_CELL,self.OnSelectCell)
        self.Bind(gridlib.EVT_GRID_CELL_RIGHT_CLICK, self.OnCellRightClick)

    def OnCellRightClick(self,evt):
        if not hasattr(self,'popupEliminarPersona'):
            self.popupEliminarPersona = wx.NewId()
            self.Bind(wx.EVT_MENU, self.OnPopupEliminarPersona, id=self.popupEliminarPersona)
        menu = wx.Menu()
        item = wx.MenuItem(menu,self.popupEliminarPersona,'Eliminar persona')
        menu.AppendItem(item)
        self.PopupMenu(menu)
        menu.Destroy()

    def OnPopupEliminarPersona(self,event):
        id = self._idSeleccionado
        row = self._rowSeleccionado
        nombre1 = self.GetTable().GetValue(row,1)
        nombre2 = self.GetTable().GetValue(row,2)
        apellido1 = self.GetTable().GetValue(row,3)
        apellido2 = self.GetTable().GetValue(row,4)

        dlg = wx.MessageDialog(self,'Desea eliminar el registro: %d %s %s %s %s'
                % (id,nombre1,nombre2,apellido1,apellido2), 'Eliminar registro',
                wx.YES_NO | wx.ICON_INFORMATION)
        if dlg.ShowModal()==wx.ID_YES:
            self.GetTable().Eliminar(id)
```

```
        dlg.Destroy()

    def OnSelectCell(self,evt):
        try:
            self._rowSeleccionado = evt.GetRow()
            self._idSeleccionado = self.GetTable().GetValue(evt.GetRow(),0)
            self._eventoSeleccionar(self._idSeleccionado)
            evt.Skip()
        except:
            evt.Skip()

class GridPersonaE(gridlib.PyGridTableBase):
    def __init__(self,parent):
        gridlib.PyGridTableBase.__init__(self)
        self._colslabels = ['ID','Primer Nombre','Segundo Nombre','Primer Apellido','Segundo Apellido','No Fotos']
        self._datatypes = [
            gridlib.GRID_VALUE_NUMBER,
            gridlib.GRID_VALUE_STRING,
            gridlib.GRID_VALUE_STRING,
            gridlib.GRID_VALUE_STRING,
            gridlib.GRID_VALUE_STRING,
            gridlib.GRID_VALUE_NUMBER
        ]
        self._data = persona.Detalle()
        self.odd = gridlib.GridCellAttr()
        self.odd.SetBackgroundColour('WHITE')
        self.even = gridlib.GridCellAttr()
        self.even.SetBackgroundColour('KHAKI')

    def GetAttr(self,row,col,kind):
        attr = [self.even,self.odd][row % 2]
        attr.IncRef()
        return attr

    def GetNumberRows(self):
        return len(self._data)+1

    def GetNumberCols(self):
        return len(self._colslabels)

    def IsEmptyCell(self,row,col):
        try:
            return not self._data[row][col]
        except IndexError:
```

```
        return True

    def GetValue(self,row,col):
        try:
            return self._data[row][col]
        except IndexError:
            return ''

    def SetValue(self,row,col,value):
        if col==0: return #no se puede modificar el id
        #modificacion de otra informacion
        try:
            self._data[row][col] = value
            r = self._data[row]
            #agregar a la BD
            id = persona.Insertar(r[0],r[1],r[2],r[3],r[4])
            self._data[row][0] = id
        except IndexError:
            #agregar una nueva fila
            self._data.append(['']*self.GetNumberCols())
            self._data[row][col] = value
            #agregar a la BD
            r = self._data[row]
            id = persona.Insertar(0,r[1],r[2],r[3],r[4])
            self._data[row][0] = id
            #indicar al grid que actualice los datos
            msg = gridlib.GridTableMessage(self,gridlib.GRIDTABLE_NOTIFY_ROWS_APPENDED,1)
            self.GetView().ProcessTableMessage(msg)

    def GetColLabelValue(self,col):
        return self._colslabels[col]

    def GetTypeName(self,row,col):
        return self._datatypes[col]

    def CanGetValueAs(self,row,col,typeName):
        colType = self._datatypes[col].split(':')[0]
        if typeName==colType: return True
        else: return False

    def CanSetValueAs(self,row,col,typeName):
        return self.CanGetValueAs(row,col,typeName)

    def Eliminar(self,id):
```

```
        view = self.GetView()
        if view:
            n = len(self._data)
            i = 0
            while i<n:
                r = self._data[i]
                if r[0]==id:
                    # eliminando de la BD
                    persona.Eliminar(id)
                    del self._data[i] #eliminando del grid
                    view.ProcessTableMessage(
                        gridlib.GridTableMessage(
                            self,gridlib.GRIDTABLE_NOTIFY_ROWS_DELETED,i,1))
                    break
                i += 1
```

## 17.1.15. Archivo: ide/res/resources.py

```
"""
A simple script to encode all the images the XRCed needs into a Python module
"""
import sys, os, glob
from wx.tools import img2py

def main():
    output = 'images.py'

    # get the list of PNG files
    files = glob.glob('*.png')
    files.append('GridBG.gif')
    files.append('python.ico')
    files.sort()

    # Truncate the inages module
    open(output, 'w')

    # call img2py on each file
    for file in files:

        # extract the basename to be used as the image name
        name = os.path.splitext(os.path.basename(file))[0]

        # encode it
```

```
        if file == files[0]:
            cmd = "-u -i -n %s %s %s" % (name, file, output)
        else:
            cmd = "-a -u -i -n %s %s %s" % (name, file, output)
        img2py.main(cmd.split())


if __name__ == "__main__":
    main()
```

## *17.2. Documentación relacionada con la Divisón Policía Técnica y Científica.*

La División Técnica y Científica de la Policía Nacional Civil ha apoyado la realización del presente proyecto con la mejor disposición; se facilitó el acceso a las instalaciones, entrevistas con el personal encargado de la realizar los restratos hablados y varias pruebas de certeza, un set de 250 rostros los cuales fueron utilizados para estimar tiempo de procesamiento así como 2 retratos hablados los cuales corroboraron la utilidad del método PCA con imágenes de este tipo.

A continuación se anexa constancia de trabajo asi como los retratos hablados que fueron utilizados en este experimento.

Se espera que este trabajo pueda servir como base para desarrollo de tecnología local útil para dicha institución.

# 18. GLOSARIO

- **Algebra Lineal**: Matemática, El álgebra lineal es la rama de la matemática que concierne al estudio de vectores, espacios vectoriales, transformaciones lineales, y sistemas de ecuaciones lineales. Los espacios vectoriales son un tema central en la matemática moderna; por lo que el álgebra lineal es usada ampliamente en álgebra abstracta y análisis funcional. El álgebra lineal tiene una representación concreta en la geometría analítica, y tiene aplicaciones en el campo de las ciencias naturales y en las ciencias sociales.

- **Algoritmo**: Esquema numérico que se usa para resolver un problema.

- **ANSI**: El American National Standards Institute es un coordinador nacional de actividades estándares voluntarias y actúa como una organización de aprobación y cámara para estándares de consenso en los EE.UU. El ANSI trabaja cercanamente con organizaciones internacionales, en particular ISO, para el desarrollo y aprobación de estándares internacionales.

- **CER (Cross-over Error Rate)**: Biométrica, Conocida también como (Equal Error Rate o EER), es la tarifa de error de cruce. Cuanto más bajo es el EER o el CER, se considera que el sistema biométrico es más exacto.

- **Correlación:** Grado de conexión entre dos objetos, más específicamente, el grado en el que un valor en un set de valores puede usarse para predecir el valor correspondiente en otro set de valores.

- **Distancia euclidea**: La distancia más corta entre dos puntos en el espacio de características, calculada usando el teorema de Pitágoras.

- **EER (Equal Error Rate)**: Biométrica, Conocida también como (Cross-over Error Rate o CER), es la tasa de error igual. Cuanto más bajo es el EER o el CER, se considera que el sistema biométrico es más exacto.

- **Eigenrostro**: Algebra Lineal, Los Eigenrostros son un set de eigenvectores usados en el problema de la visión computarizada para el reconocimiento de rostros humanos. Estos eigenvectores son derivados de la matriz de covarianza y provienen de la distribución probabilística del mas alto vector dimensional entre el espacio de vectores posibles, resultantes de los rostros humanos.

- **Eigenvalor**: Algebra Lineal, Es el escalar $\lambda$ y define la magnitud del eigenvector, recibe el nombre autovalor o valor característico o eigenvalor.

- **Eigenvector**: Algebra Lineal, Los autovectores o eigenvectores de un operador lineal son los vectores no nulos que, cuando son transformados por el operador, dan lugar a un múltiplo escalar de sí mismos, con lo que no cambian su dirección.

- **FAR**: Biométrica, Tasa de falso positivo (False Acceptance Rate o FAR), Porcentaje de incidentes falsos que fueron aceptados como válidos.

- **FER**: Biométrica, Tasa de fallo en alistamiento o enrolamiento, Porcentaje de incidentes verdaderos que no fueron enrolados o enlistados correctamente.

- **FNMR**: Biométrica, Tasa de Incidentes no Encontrados, Porcentaje de incidentes verdaderos que no fueron encontrados.

- **FRR**: Biométrica, Tasa de falso negativo (False NonMatch Rate o FNMR y el fallo de tasa de alistamiento (Failure-to-enroll Rate, FTR o FER). Porcentaje de incidentes verdaderos que fueron rechazados como inválidos.

- **ICA**: Algebra Lineal, Análisis de componentes independientes. Método que consiste en extraer las características independientes de los objetos en los espacios y subespacios Vectoriales.

- **Identificación**: Procedimiento de reconocimiento de la identidad y características de un objeto/sujeto.

- **Matriz de Covarianza**: Algebra Lineal - Matemática, Matriz resultado del análisis de covarianza, de carácter estadístico y que basa su proceso en una técnica estadística que, utilizando un modelo de regresión lineal múltiple, busca comparar los resultados obtenidos en diferentes grupos de una variable

cuantitativa pero corrigiendo las posibles diferencias existentes entre los grupos en otras variables que pudieran afectar también al resultado (covariantes).

- **Open Source**: Informática, Código abierto (del inglés open source) es el término por el que se conoce al software distribuido y desarrollado en forma libre. Este término empezó a utilizarse en 1998 por algunos usuarios de la comunidad del software libre, tratando de usarlo como reemplazo al ambiguo nombre original en inglés del software libre (free software).

- **PCA**: Algebra Lineal, El Análisis de componentes principales (PCA) es empleado como técnica de extracción de características para reducir la dimensión de los modelos o clases de objetos.

- **Prototipo**: Ciencia, Cualquier tipo de máquina en pruebas, o un objeto diseñado para una demostración de cualquier tipo. Este tipo de prototipos permiten testar el objeto antes de que entre en producción, detectar errores, deficiencias, etcétera. Cuando el prototipo está suficientemente perfeccionado en todos los sentidos requeridos y alcanza las metas para las que fue pensado, el objeto puede empezar a producirse.

- **Python**: Informática, Python es un lenguaje de programación interpretado e interactivo, capaz de ejecutarse en una gran cantidad de plataformas. Fue creado por Guido van Rossum en 1990. Python es habitualmente comparado a TCL, Perl, Scheme, Java y Ruby. Actualmente, Python se desarrolla como un proyecto de código abierto, administrado por la Python Software

Foundation. La última versión estable del lenguaje es actualmente (Marzo de 2006) la 2.4.3. Guido van Rossum, más conocido como Guido, creó Python, un lenguaje de programación de scripting, la "oposición leal" a Perl, lenguaje con el cual mantiene una rivalidad amistosa. Los usuarios de Python consideran a éste mucho más limpio y elegante para programar. Python es un lenguaje interpretado, lo que ahorra un tiempo considerable en el desarrollo del programa, pues no es necesario compilar ni enlazar. El intérprete se puede utilizar de modo interactivo, lo que facilita experimentar con características del lenguaje, escribir programas desechables o probar funciones durante el desarrollo del programa. También es una calculadora muy útil.

- **Sistemas biométricos**: Biométrica, Entenderemos por sistema biométrico a un sistema automatizado que realiza labores de biometría. Es decir, un sistema que fundamenta sus decisiones de reconocimiento mediante una característica personal que puede ser reconocida o verificada de manera automatizada.

- **Visión Artificial**: Informática, la Visión artificial, también conocida como Visión por Computador (del inglés Computer Vision) o Visión técnica, es un subcampo de la inteligencia artificial. El propósito de la visión artificial es programar un computador para que "entienda" una escena o las características de una imagen.