Sistema de Autentificación Facial

Mikaela Bauzá, Joaquin Vanschoren•, Marcelo P. Funes, Gabriel M. Barrera, Daniela López De Luise

Abstract

El presente trabajo presenta un framework llamado AURA, basado en el modelo de Codelets para la validación de imágenes faciales contra cierto perfil de características previamente almacenado. Cada imagen se descompone generando eigenfaces, que permiten reducir la cantidad de información necesaria para el almacenamiento y procesamiento del rostro. Dichos eigenfaces son almacenados para luego ser utilizados en la comparación contra una nueva imagen. El sistema permite validar rostros aún con variaciones de iluminación, cambios de ubicación, cambios físicos y estéticos entre otras cosas.

Palabras claves: Codelets, eigenfaces, autenticación facial, biometría, Flexo.

• J. Vanschoren es autor del framework Flexo, y se desempeña en la Universidad de Leuven, en Bélgica.

1. Introducción

El estudio de validación de identidad por medios automáticos basados en uno o más rasgos físicos es conocido como biometría y es muy investigado. Actualmente se han hallado diversas formas de validar la identidad de una persona, por ejemplo por medio de la comparación del patrón del iris, características específicas de las huellas digitales, reconocimiento de las frecuencias en la voz, comparación de algunas características en el rostro, etc. [1].

La investigación para autentificar por medio del rostro tiene más de veinte años de trabajo, en los que se han desarrollado diferentes algoritmos y técnicas. Uno de los mecanismos más comunes es utilizar Redes Neuronales (RN) para obtener un conjunto de características denominadas eigenfaces. Estas eigenfaces son útiles reducir la cantidad de procesamiento necesario y mejorar la precisión de los resultados. Por caso se puede citar el trabajo de Renato Salinas y Luís Larraguibel [2], donde se usa una red neuronal con arquitectura paramétrica para el reconocimiento de personas.

También es destacable el desarrollo de la empresa F7 Corporation (Instituto Biométrico de Reconocimiento Facial) [3] para la construcción de un software de biometría. El sistema está planteado básicamente para entidades financieras garantizando la seguridad de los clientes, ya que se identifica automáticamente a cada uno de ellos a través de una imagen digital de su cara en movimiento mediante una cámara web.

La empresa Google desarrolló un producto llamado Picasa [4] y una herramienta de búsqueda de reconocimiento de rostros. El software cuenta con una interfase que permite subir álbumes de fotos en Internet para incorporarla a un banco de imágenes. La herramienta busca e identifica automáticamente el rostro de cualquier persona que le indiquemos, y muestra todas las imágenes donde crea que ésta se encuentre.

En el caso de BananaScreen, el software de reconocimiento de rostro está diseñado para una computadora personal. Con él se puede añadir una capa más de protección al sistema operativo, configurándolo para que se bloquee automáticamente cada cierta cantidad de minutos y activarlo nuevamente utilizando el rostro [5].

Bioscrypt es otro proveedor de esta tecnología de control de acceso y desarrolló un método de reconocimiento llamado Bioscrypt VisionAccess 3D Face Readers [6]. Para este sistema se necesita instalar una cámara especial. El proceso de reconocimiento consta de tres etapas. En la primera se hace la captura del rostro usando una luz cercana al rango del infrarrojo. El sistema proyecta un patrón de luz invisible en la cara, el cual se distorsiona por la geometría de la superficie del rostro y la cámara capta esta distorsión. Luego, se reconstruye en tiempo real la superficie facial en tres dimensiones, la cual no es almacenada en la base de datos. Por

último, se extrae una plantilla biométrica de la geometría facial 3-D y la verificación se realiza comparando esta plantilla con la que está almacenada en una tarjeta inteligente.

En el presente trabajo se presenta un prototipo destinado a lograr la autenticación facial. A diferencia de los casos mencionados anteriormente, este algoritmo en principio sería más robusto puesto que, además de superar los clásicos problemas de iluminación, nitidez, resolución y centrado de los rostros, es capaz de validar aún habiendo cambios físicos o estéticos en el rostro entre otras cosas, utilizando eigenfaces y Codelets implementados sobre un framework llamado Flexo [7].

2. Eigenfaces

Los eigenfaces son vectores de datos utilizados en computación para el reconocimiento facial, a los que se los denomina eigenvectores, donde cada uno posee valores escalares conocidos como eigenvalores. De la misma forma que cualquier color puede ser creado por la mezcla de los colores primarios, cualquier cara podría generarse mediante una combinación de eigenfaces.

El enfoque de la utilización de eigenfaces fue desarrollado por Sirovich y Kirby en 1987 [8] y utilizado por Matthew Turk y Alex Pentland en la clasificación de rostros [9]. Estos últimos presentaron un proyecto que permite el reconocimiento de la cara en tiempo casi real aprovechando el hecho que generalmente los rostros se encuentran en posición vertical y, por lo tanto, pueden ser descritos como un grupo de características vistas en dos dimensiones [10].

Para generar un conjunto de eigenfaces se debe digitalizar un grupo de imágenes de rostros tomadas bajo las mismas condiciones de luz, luego se normalizan a la línea de los ojos y de la boca.

Los eigenfaces pueden extraer datos por medio de una herramienta matemática conocida como Análisis de Componentes Principales (Principal Component Analysis, PCA) [11]. Los eigenfaces creados aparecerán como zonas luminosas y oscuras organizadas con un patrón específico. Este patrón son las diferentes características de la cara que están señaladas para ser evaluadas, por ejemplo habrá un patrón para evaluar la simetría, otro para determinar el tamaño de la nariz o la boca, etcétera. Algunos eigenfaces tienen patrones más difíciles de identificar y el resultado final puede parecerse muy poco a un rostro.

Se necesita seguir una serie de pasos para la creación de los de eigenfaces. Primero se debe preparar y procesar el set de entrenamiento, las imágenes tienen que tener la misma resolución y los rostros deben estar alineados. Cada imagen es vista como un vector, concatenando las filas de píxeles de la imagen original y se debe asumir que las imágenes de la serie de formación se almacenan en una única matriz, donde cada fila es una imagen. Luego, el promedio de cada imagen tiene

que ser calculado y se lo resta a la imagen original que está en la matriz. También se deben computar los eigenvectores y eigenvalores de la matriz de covarianza, donde cada vector tiene la misma dimensión que la imagen original y estos eigenvectores son los llamados eigenfaces. Por último se eligen los componentes principales, la matriz NxN de covarianza se transformará en N eigenvectores donde cada uno representa un punto en el espacio.

Después de realizar esta serie de pasos y cálculos habrá miles de eigenfaces creados, pero los realmente necesarios son muchos menos, por lo que hay que seleccionar los que posean eigenvalores más altos.

En la actualidad hay algunos software que se encargan de realizar este proceso, uno de ellos es el FreeMat [12] que es gratuito, y además es *open source*.

3. Codelets

Los Codelets son agentes computacionales no determinísticos que transportan pequeñas piezas de código. Estos pueden ser llamados por cualquier entidad que note la necesidad de ejecutarlos, aunque sea otro Codelet, ya que son los responsables de proponer nuevas estructuras o evaluar las estructuras propuestas para generar nuevos Codelets, los cuales continuarán el proceso en la próxima etapa.

Los que tiene altas probabilidades de ser seleccionados para correr forman el Coderack, de donde entran y salen en forma pseudo aleatoria y quedan ahí hasta que se dispara su ejecución, ya que cada uno tiene un valor de urgencia asignado y son seleccionados para correr de acuerdo a este valor [13].

Todo el proceso se produce a través de las acciones colectivas de muchos Codelets que trabajan en paralelo, a diferentes velocidades, en diferentes aspectos del problema, sin ningún tipo de control centralizado sobre el curso de los acontecimientos.

Generalmente los Codelets son comparados con una colonia de hormigas, donde el trabajo de cada una en particular puede ser pequeño o insignificante pero al trabajar en conjunto llevan adelante una gran tarea donde se auto controlan.

En el proyecto Copycat de Douglas Hoftadter y Melanie Mitchell [14] se implementó el modelo de Codelets, que luego utilizó James B. Marshall para desarrollar Metacat [13] [15] como una versión mejorada del proyecto. También Stan Franklin y su grupo de investigadores desarrollaron un modelo más complejo basado en este esquema y en la *global workspace theory*, desarrollada por Bernard Baars [16]. Uno de los proyectos donde utilizaron esta técnica es IDA (Intelligent Distribution Agent) [17] que asigna personal para la marina de los Estados Unidos de acuerdo a sus preferencias y a las políticas de esta fuerza de seguridad.

4. Flexo

Flexo es un ambiente experimental de conceptos fluidos [7]. Este framework trabaja con Codelets, por lo cual los conceptos se manejan como entidades en una red de fluidos, capaces de adaptarse a diferentes situaciones a través del aumento y la redistribución de activación. El programa utiliza las presiones emergentes de la situación para hacer una interpretación, si no es la esperada se modifica hasta que encaje nuevamente. No hay un control central, pero existen pequeños mecanismos y muchos agentes que trabajan en paralelo para la creación de las presiones que guían el proceso hasta alcanzar una aceptación. El proyecto AURA se desarrolla como una extensión y especialización de este framework original.

4.1. Relación entre Copycat y Flexo

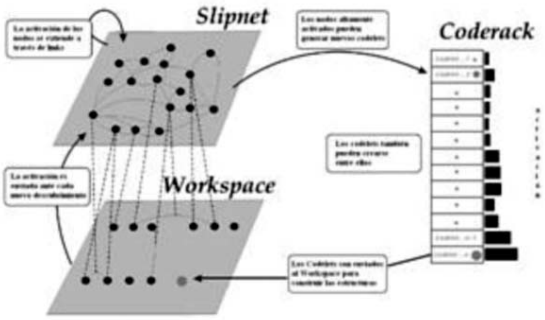
El sistema provee los mecanismos comúnmente usados en las implementaciones de conceptos fluidos, ya que está basado en el Copycat, en donde se plantea básicamente una analogía entre un modelo computacional de alto nivel de percepción y el modelo de mecanismos fundamentales subyacentes al conocimiento humano.

El dominio de Copycat son las 26 letras minúsculas del abecedario inglés y la analogía de los problemas está basada en tres cadenas de caracteres: la inicial, la modificada y la esperada. En el Copycat las letras son cosas abstractas y las únicas relaciones posibles entre ellas están definidas por ser la misma letra, la predecesora o la sucesora.

El poder de este dominio está en la habilidad de modelar diferentes aspectos del mundo real con estructuras abstractas comunes. Esa es la idea principal que toma Flexo para resolver sus problemas, teniendo como ejemplo principal un Copycat desarrollado entre sus clases.

4.2. Componentes de Flexo

El framework cuenta con cuatro componentes básicos: el Slipnet, el Workspace, el Coderack y los Codelets; proporcionando los medios necesarios para entrar en la estructura del dominio, a través de objetos relacionados por medio de links. La interacción entre los distintos componentes se ve reflejada en la figura 1.



cambiar

Figura 1 | **Componentes de Flexo**

El Slipnet hace que el usuario pueda tener una idea de nodos, enlaces y todas sus propiedades, ya que es una red de conceptos o nodos conectados por relaciones conceptuales conocidos como links. Los nodos poseen un grado de activación, que puede subir, reflejando la importancia del concepto en el problema. Cuando la activación llega a un tope, el concepto se dispara. Con el tiempo la activación disminuye gradualmente, si el concepto no es nuevamente invocado. Los links pueden cambiar la distancia conceptual, marcando así la relación entre dos conceptos.

El Workspace permite definir las estructuras que también pueden ser construidas por Codelets, las estructuras comunes tienen que ser reutilizables y modificables. En un principio, se presentan un gran número de equipos de constructores que tienen todos el mismo conjunto de bloques, es decir, el concepto básico del problema. Luego van cambiando para construir estructuras más elegantes, empezando en diferentes momentos pero trabajando en paralelo. Las estructuras varían hasta ser lo suficientemente convincentes, o sea, se asemejan a los conceptos que se encuentran en el Slipnet. Cabe aclarar que estas nuevas estructuras pueden resultar de la unión o división de estructuras más grandes.

El módulo de Coderack es el lugar donde está el conjunto de algoritmos de control, que además debe admitir la carga de Codelets definidos por el usuario, y todos sus parámetros deben ser fácilmente modificables. Los Codelets realizan pequeñas acciones en el Workspace, manteniendo siempre su jerarquía. Hay algunas clases básicas de Codelets donde cada una es más apropiada para una estructura en particular, aunque cada usuario puede crear sus propios tipos de Codelets para el problema específico que esté tratando. Los módulos de auto control deben ser fáciles de añadir y usar, ya que son los encargados de vigilar el comportamiento del sistema.

Con todo esto el Slipnet puede ser visto como una memoria a largo plazo y el Workspace, es entonces, una memoria de corto plazo donde se trata el problema

actual; por lo tanto, debe existir un ida y vuelta entre todos los componentes para distinguir lo que realmente es útil y probar continuamente su calidad, sobretodo entre la actividad perceptiva (Workspace) y la actividad conceptual (Slipnet).

5. AURA

Aura es un prototipo que resulta de la investigación sobre validación de la imagen de un rostro contra un perfil de características previamente almacenado.

Este sistema de validación facial consta básicamente de tres bloques: preproceso, extracción de características y clasificación, como se muestra en la figura 2.

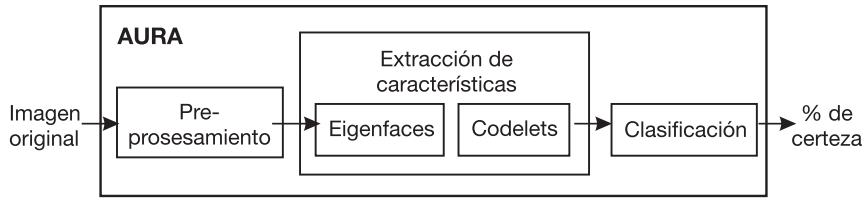


Figura 2 | **Esquema de AURA**

En la primera etapa de pre-procesamiento, se trata de restaurar y normalizar las imágenes de entrada, a fin de lograr una buena localización y una buena calidad de imagen. La restauración aplica un filtro para reducir el ruido generado por el dispositivo de captura, en cambio, la normalización aplica una corrección para aclarar las imágenes con baja luminosidad. El segundo paso es extraer las características realizando las tareas necesarias para localizar el rostro y luego parametrizar la imagen facial. Utilizando eigenfaces se obtiene un vector característico representativo del rostro que ayuda a reducir la cantidad de información necesaria para el almacenamiento. Por último, se efectúa la clasificación, comparando el nuevo patrón con los patrones de usuarios registrados y almacenados en una base de datos, verificando que la identidad reconocida corresponda a la ingresada [18]. Cabe aclarar que el sistema determina que el rostro corresponda al buscado con un cierto grado de confianza.

Para este último paso se utiliza una adaptación y extensión del framework Flexo, ya que entre otras cosas se lo modificó para que esté preparado para futuros cambios, por ejemplo utilizando genéricos para el lenguaje Java 1.6 y definiéndole los Codelets adecuados al tema. El sistema por medio de estos agentes computacionales compara la imagen dada con las imágenes almacenadas, dando así una respuesta al problema que se está tratando.

El sistema cuenta con dos pantallas principales para que el usuario pueda utilizarlo, como se ve a continuación en la figura 3. En las imágenes se muestra la pantalla principal donde se captura la foto del usuario para luego realizar la validación. En ambos casos, tanto la captura como la validación, pueden ser canceladas por el usuario. También hay una pantalla de configuración, desde donde se puede seleccionar el dispositivo de captura.

Figura 3 | **Captura y validación de la imagen**

Cuando se instala el programa se crea un usuario administrador que tiene autorización para agregar/capturar las imágenes que forman la base de datos. Para poder utilizar este sistema sólo se necesita tener una computadora personal y una cámara web. El programa corre bajo Windows pero está preparado para que sea transportable a otros sistemas operativos.

6. Trabajo futuro

El prototipo actualmente está en plena incorporación de conceptos. Se está integrando una interfase apropiada para unificar el preprocesamiento al resto del sistema.

Dado que el sistema deberá ser manejado por usuarios principiantes que posean mínima capacitación técnica, como fotografiar a una persona y correr una aplicación visual, será necesario simplificar las interfases actuales que muestran parte de la actividad interna, de modo que éstas sean trasparentes al usuario.

También, se estudiará la posibilidad de incorporar una herramienta adicional que permita reformular dinámicamente los parámetros del sistema, de manera que los mismos puedan ser adaptados a distintos dominios.

Finalmente, queda por realizar pruebas de carga, estrés y comparar los resultados contra otras alternativas del mercado, con fines estadísticos.

Referencias

[1] Fisher R. A., “Biometry”. Biometrics vol. 4: 217-219. (1948).

[2] Salinas R., Larraguibel L., “Red Neuronal de Arquitectura Paramétrica en Reconocimiento de Rostros”. Universidad de Chile, Chile.

[3] F7 Corporation (Instituto Biométrico de Reconocimiento Facial), http://www.f7corp.com

[4] Picasa, http://www.picasa.com

[5] Banana Security, BananaScreen, http://www.bananasecurity.com

[6] Bioscrypt, VisionAccess 3D Face Readers, http://www.l1id.com

[7] Vanschoren J. “Development of a framework for high-level perception”. Master’s thesis, Dept. of Computer Science, Katholieke Universiteit Leuven, Leuven, Belgium. http://www.cs.kuleuven.be/~joaquin/Flexo (2005)

[8] Sirovich L., Kirby M., “Low-dimensional procedure for the characterization of human faces”. Journal of the Optical Society of America, vol. 4: 519–524. (1987).

[9] Turk M., Pentland A., “Eigenfaces for recognition”. Journal of Cognitive Neuroscience, vol. 3: 71–86. (1991).

[10] Turk M., Pentland A., “Face recognition using eigenfaces”. Proc. IEEE Conference on Computer Vision and Pattern Recognition: 586–591. (1991).

[11] Moon H., Philips P. J., “Computational and performance aspects of PCA-based face-recognition algorithms”. Perception, 30: 303-321. (2001)

[12] FreeMat Team, FreeMat, http://freemat.sourceforge.net

[13] Marshall J. B., “Metacat: A Self-Watching Cognitive Architecture for Analogy-Making and High-Level Perception”. Indiana University, USA. 1999.

[14] Hofstadter D. R., Mitchell M., “The Copycat Project: A Model of Mental Fluidity and Analogy-Making”. Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought, 31-112. Basic Books. 1995.

[15] Marshall J. B., Hofstadter D. R., “The Metacat Project: A Self-Watching Model of Analogy-Making”. Indiana University, USA.

[16] Baars B., “A cognitive theory of consciousness”. NY: Cambridge University Press. 1988.

[17] Franklin S., Kelemen A., McCauley L. “IDA: A Cognitive Agent Architecture”. IEEE Conf on Systems, Man and Cybernetics. 1998.

[18] Long S., Müller O. V., “Verificación biométrica automática de identidad mediante reconocimiento facial”. Tesis de grado. Facultad de Ingeniería y Ciencias Hídricas, Universidad Nacional del Litoral. Argentina. 2006.