**UNIVERSIDAD AUTÓNOMA DE MADRID**

**ESCUELA POLITÉCNICA SUPERIOR**

** **

**Grado en Ingeniería Informática**

**TRABAJO FIN DE GRADO**

**Framework orientado a algoritmos de recomendación basados en vecinos cercanos**

**Alejandro Gil Hernán**

**Tutor: Nombre y Apellidos**

**Ponente (si procede): Nombre y Apellidos**

**Febrero 2017**

**Framework orientado a algoritmos de recomendación basados en vecinos cercanos**

**AUTOR: Alejandro Gil Hernán**

**TUTOR: Nombre y Apellidos tutor/a**

**Grupo de la EPS (opcional)**

**Dpto. XXX**

**Escuela Politécnica Superior**

**Universidad Autónoma de Madrid**

**Febrero 2017**

**Resumen (castellano)**

Los sistemas de recomendación se han convertido en una pieza clave en internet, bien sea navegando a través de artículos científicos (Mendeley), música (Spotify, Last.fm), películas o vídeos (Netflix, IMDB, YouTube), personas (LinkedIn, Facebook, Twitter), así como plataformas online de venta de infinidad de productos (Amazon).

Dichos sistemas de recomendación están motivados a su vez por el crecimiento exponencial que ha experimentado la web en los últimos años y con la aparición de gran contenido digital y se caracterizan por observar la actividad de los usuarios y aprovecharla para predecir cuáles son los intereses de éstos, según los cuales se presentarán unos productos u otros, de una forma individual y personalizada.

En este TFG se explorará un tipo de algoritmo de recomendación muy habitual: vecinos cercanos. Estos algoritmos se suelen utilizar para hacer recomendaciones basadas en similitudes entre usuarios o entre objetos, siendo de esta forma un tipo de filtrado colaborativo; sin embargo, si la similitud tiene en cuenta atributos de los usuarios o de los objetos, también se podría utilizar para algoritmos basados en contenido. Este potencial permite, en principio, que se puedan utilizar en multitud de dominios, teniendo la ventaja adicional de que su salida es fácil de interpretar y analizar.

En este trabajo se plantea diseñar e implementar un framework orientado a generar y evaluar recomendaciones basadas en este tipo de algoritmos, así como el estudio sobre los distintos parámetros a configurar para discernir cuál es la combinación que aporta mejores resultados.

Por ello los objetivos principales serán obtener implementaciones generales, así como que se ejecuten de la forma más eficiente posible.

**Abstract (English)**

Recommender systems have become a key element in the internet in different aspects such as scientific articles browsing (Mendeley), music (Spotify, Last.fm), movies (Netflix, IMDB), people (LinkedIn, Facebook, Twitter), even online selling platforms (Amazon). Those recommender systems are also motivated by the exponential growth experimented by the web in the past years and with the appearance of substantial quantity of digital content and they are characterised by taking advantage of observing user´s activity to predict their interests. This results will be presented differently each time in an individual and personalised way.

In this bachelor thesis, it will be explored a very common kind of recommendation algorithm: k-nearest neighbours (KNN). These algorithms are usually used to make recommendations based on similarities between users or items, being that way a kind of collaborative filtering; however, if the similarity considers user´s or items attributes, it also will be possible to use it for content-based algorithms. This potential allows to use them in many domains, having the advantage of the easily interpretable and analysable output.

This work contemplates to design and implement framework oriented to generate and evaluate recommendations based on these kind of algorithms, as well as the study of the different parameters configurations to distinguish which is the most beneficial combination.

Therefore, the principal objectives will be to obtain general implementations which will be executed in the most efficient way.

**Palabras clave (castellano)**

Sistema de recomendación, algoritmo, framework, vecinos cercanos (KNN), filtrado colaborativo, algoritmo basado en contenido.

**Keywords (inglés)**

Recommender system, algorithm, framework, nearest neighbours (KNN), collaborative filtering, content-based.

***Agradecimientos***

*Quiero agradecer en primer lugar a mi familia, sobre todo a mis padres y abuelos.*

*A mi madre por criarme y hacer de mí la persona que soy.*

*A mi padre por sus consejos.*

*A mis abuelos por tratarme como a un hijo.*

*A mi tutor Alejandro por guiarme en este trabajo y asignaturas de la carrera, por ser un profesor ejemplar con una dedicación como jamás he visto en el mundo docente.*

*Muchas gracias a mis amigos que son como hermanos y a Paula por estar siempre apoyándome.*

*Por último, dar las gracias a la Escuela Politécnica Superior por hacerme conocer a compañeros increíbles y formarme como profesional y como persona.*

**Alejandro Gil Hernán**

**ÍNDICE DE CONTENIDOS**

[1 Introducción 1](#_Toc471136422)

[1.1 Motivación 1](#_Toc471136423)

[1.2 Objetivos 2](#_Toc471136424)

[1.3 Organización de la memoria 3](#_Toc471136425)

[2 Estado del arte 3](#_Toc471136426)

[2.1 Tipos de algoritmos 4](#_Toc471136428)

[2.1.1 Algoritmos basados en contenido. 4](#_Toc471136429)

[2.1.2 Algoritmos basados en filtrado colaborativo. 4](#_Toc471136430)

[2.1.3 Algoritmos generales. 6](#_Toc471136431)

[3 Diseño 9](#_Toc471136432)

[3.1 Subsección 9](#_Toc471136433)

[3.1.1 Subsubsección 9](#_Toc471136434)

[4 Desarrollo 10](#_Toc471136435)

[4.1 Subsección 10](#_Toc471136436)

[4.1.1 Subsubsección 10](#_Toc471136437)

[5 Integración, pruebas y resultados 11](#_Toc471136438)

[6 Conclusiones y trabajo futuro 11](#_Toc471136439)

[6.1 Conclusiones 11](#_Toc471136440)

[6.2 Trabajo futuro 11](#_Toc471136441)

[Referencias 13](#_Toc471136442)

[Glosario 15](#_Toc471136443)

[Anexos I](#_Toc471136444)

[A Manual de instalación I](#_Toc471136445)

[B Manual del programador III](#_Toc471136446)

[C Anexo … - 1 -](#_Toc471136447)

**INDICE DE FIGURAS**

[Figura 1: Valor de las ventas del comercio electró 3](#_Motivación)

**INDICE DE TABLAS**

# Introducción

Los sistemas de recomendación (SR) son herramientas software y técnicas que proveen al usuario de elementos que pueden resultarle interesantes y afines a sus gustos, facilitando la labor de búsqueda entre la inmensa cantidad de información de la que dispone. Éstas recomendaciones están relacionadas con las acciones que lleva a cabo el usuario: productos comprados, música que escucha, libros que lee… etc.

Hoy en día estamos más que acostumbrados a que prácticamente la totalidad de las aplicaciones informáticas de uso diario nos propongan contenido, ya sea porque Amazon nos recomiende un producto que comprar, Spotify una canción que escuchar o Youtube un vídeo que quizá nos guste. En definitiva, no nos resulta extraño que la propaganda *online* se adapte a nosotros con recomendaciones personalizadas.

## Motivación

Los sistemas de recomendación nacen con el objetivo de facilitar la toma de decisiones al usuario, así como de aumentar el beneficio de empresas dedicadas al comercio *online*. Desde la aparición de la WWW (world wide web) el comercio en internet ha experimentado un crecimiento exponencial, hasta el punto que la comisión nacional de los mercados y la competencia (CNMC) determinó que la facturación de las tiendas *online* en España crecía a un 25% interanual.



**Figura 1. Valor de las ventas del comercio electrónico en EEUU**

Para conseguir éstos buenos resultados, los puntos más importantes de los SR son la valoración y comparación de los resultados obtenidos en las recomendaciones, lo que se conoce como evaluación. Durante mucho tiempo la manera natural de evaluar tanto buscadores como recomendadores eran las métricas de error, pero es evidente que a lo largo de los años las tendencias varían, por ello, en la actualidad, están evolucionando hacia métricas basadas en ranking, ampliando un nuevo horizonte de estudio y desarrollo. Ahora queda pendiente la comparativa de efectividad entre los distintos algoritmos y variantes dentro de éstos con este nuevo tipo de métricas.

Este trabajo de fin de grado se ha realizado con la intención de crear un framework donde sea posible la comparación de resultados para recomendaciones generadas con un algoritmo basado en vecinos cercanos (KNN) y todas sus variantes posibles, ya que como se ha mencionado anteriormente, esto viene motivado por el cambio en las métricas de evaluación (la manera en que se mide la tasa de error y, por tanto, la eficacia en una recomendación).

Esto es importante ya que los sistemas de recomendación ofrecen una serie de ventajas a las plataformas de productos online:

* Incrementar el número de ítems vendidos: ya que a cada usuario se le muestra primero lo que potencialmente más le atrae.
* Aumentar la diversidad: Una característica es generar diversidad en las recomendaciones, sugiriendo al usuario ítems similares a sus gustos con un índice de popularidad menor.
* Incrementar la satisfacción del usuario.
* Incrementar la fidelidad del usuario.

Es por esto que el método de evaluación en este trabajo tendrá presente las tres métricas que creemos más influyentes actualmente: precisión, recall y NDCG (Normalized Cumulative Discounted Gain).

## Objetivos

La meta general de este trabajo es averiguar cuál de las configuraciones del algoritmo KNN basado en filtrado colaborativo implementadas arroja mejores resultados.

De forma particular, los objetivos a llevar a cabo son los siguientes:

* Fijar el punto de partida con un dataset concreto (no se va a tratar la recogida de información, por lo que se supone que ésta se obtiene de forma externa al proyecto).
* Estudio de las variantes del algoritmo KNN, así como de sus parámetros de entrada, con cada uno se realizará una variación.
* Estudio de métodos alternativos para el cálculo de vecinos, sin basarse en el procedimiento tradicional de los SR y aumentando la eficiencia.
* Generar recomendaciones para los usuarios del sistema con la información del dataset inicial.
* Evaluar las recomendaciones generadas con las tres métricas más importantes mencionadas anteriormente.
* Sintetizar los resultados en una tabla comparativa, pudiendo escoger cualquier combinación deseada.

Para contrastar las diferentes variaciones simplemente será necesario buscar en la tabla comparativa de resultados la combinación deseada, que mostrará los valores de las distintas métricas de error.

## Organización de la memoria

La memoria consta de los siguientes capítulos:

* **…**

# Estado del arte

En la última década, los sistemas de recomendación están adquiriendo una importancia esencial debido al crecimiento del comercio electrónico y a la repercusión que Internet está teniendo en los consumidores.

Es común asociar el mundo de los SR al de buscadores, aunque en realidad son algo dispares: mientras que en un buscador es necesario que el usuario introduzca explícitamente lo que quiere buscar, un recomendador actúa como una entidad activa, recuperando información de manera implícita a través de las interacciones de los usuarios y sugiriendo ítems de una manera transparente.

A su vez, los SR comparten cualidades con el aprendizaje automático, ya que el sistema también aprende los intereses y gustos del usuario detectando patrones de comportamiento. Varias técnicas de los SR como el cálculo de vecinos, la factorización de matrices o las particiones del dataset utilizan conceptos y herramientas conocidas en ambos campos.

Los usuarios de un SR pueden tener características muy distintas; no obstante, para poder realizar recomendaciones personalizadas, lo esencial es estudiar la estructura que tiene la información que se posee. Ésta información puede estar estructurada de varias maneras, la selección de qué atributos son relevantes depende de la técnica de recomendación que se vaya a utilizar. Por ejemplo, en filtrado colaborativo, los usuarios son estructurados como una lista que contiene los diferentes ítems que ese usuario ha puntuado, aunque también pueden ser descritos por su patrón de comportamiento, como las acciones que llevan a cabo a la hora de navegar, por ejemplo, dónde se hace click, de forma que es posible conocer los intereses también de esta manera.

Aun así, las puntuaciones o *ratings* son la forma más común de recoger los datos en un SR. Pueden ser obtenidos de manera implícita (el número de veces que se escucha una canción) o explícita (la puntuación numérica que un usuario da a un ítem concreto).

Existen diferentes tipos de ratings explícitos:

* Numéricos: puntuación normalmente comprendida entre 1 y 5.
* Binarios: me gusta, no me gusta
* Ordinales (en desacuerdo, de acuerdo, muy de acuerdo…).

Por otro lado, los tipos de ratingsimplícitos dependen de la plataforma donde esté implementado el SR, por ejemplo, en el caso de canciones, el rating representa las veces que ha sido escuchada cada canción o cada artista; para productos, puede representar simplemente el haber hecho click, y para vídeos el tiempo de visualización, etc. Con este tipo de ratings, el hecho de que un usuario no haya interaccionado con un ítem concreto normalmente se entiende como que no conoce dicho ítem, mientras que, si el valor es muy bajo, se interpreta como que tiene poco interés en el mismo. En cambio, la ausencia de ratings explícitos aporta un matiz diferente a la hora de recomendar, ya que el usuario puede conocer el ítem, pero prefiere no invertir ese tiempo en dar dicha información al sistema.

Un evento a destacar ocurrido en los últimos años ha sido el *Netflix prize* (premio Netflix), dónde la famosa empresa multinacional de servicios multimedia (series y películas) propuso una competición abierta en busca del mejor algoritmo de filtrado colaborativo para generar recomendaciones de películas. En el reto podía participar cualaquier persona excepto aquellas cercanas a la propia empresa.

El 21 de septiembre de 2009 el gran premio de US$1.000.000 fue otorgado al equipo de BellKor's Pragmatic Chaos, formado por la unión de los equipos Bellkor in BigChaos and Pragmatic Theory, quienes consiguieron una mejora de 10,06% sobre el algoritmo de Netflix.

Este reto es importante ya que junto a él surge la motivación de mejorar y optimizar los métodos de recomendación, así como la publicación de un dataset muy amplio y relevante a nivel mundial con 480.000 usuarios, 17.000 películas y más de 100 millones de ratings.

## Tipos de algoritmos

Dependiendo de la forma en que los algoritmos tengan en cuenta los atributos de los ítems o usuarios, podemos concluir que existen dos tipos de algoritmos principales: basados en contenido y basados en filtrado colaborativo.

## 2.1.1 Algoritmos basados en contenido.

Cuando se aplican métodos basados en contenido, el sistema aprende a recomendar ítems similares a los que le han gustado al usuario en un pasado. Esta similitud es calculada basándose en las características asociadas a los ítems. En el caso de que un usuario haya puntuado con buena nota una película cuyo género es comedia, es muy probable que se generen recomendaciones de películas con este mismo género. En el caso de ítems más complejos como documentos, se calcula el *TF -IDF* (Term Frequency-Inverse Document Frequency) que comprende las palabras o términos más relevantes.

Los algoritmos más comunes de éste género son *Rocchio* y KNN.

## 2.1.2 Algoritmos basados en filtrado colaborativo.

A diferencia de los métodos basados en contenido, el filtrado colaborativo se basa en los ratings de los usuarios del sistema. La idea clave es que, si dos usuarios *u* y *v* son similares, el sistema recomendará a *u* ítems que le hayan gustado a *v*, ya que en teoría sus gustos son similares.

El filtrado colaborativo soluciona algunas de las limitaciones de los algoritmos basados en contenido. Por ejemplo, los ítems cuyo contenido es difícil de obtener pueden seguir siendo recomendados a usuarios por la similitud con otros. Otra ventaja es que el sistema puede recomendar mucha más variedad de elementos, nunca produciéndose el encasillamiento que ocurre con los basados en contenido, ya que las tendencias de una comunidad de usuarios pueden ir variando, a la vez que el SR se adapta a ellas.

***Basados en usuario (User KNN)***

Se recomiendan al usuario los ítems que han gustado a usuarios similares a éste.



***Basado en ítem (Item KNN)***

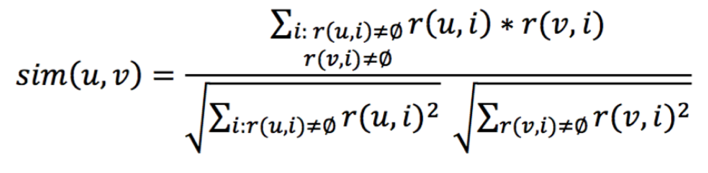
Se recomiendan al usuario los ítems que se parecen a ítems que le han gustado.



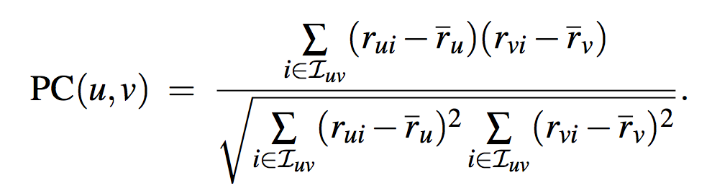
Los algoritmos KNN requieren de un vecindario para realizar el cálculo de ratings. Si se aplica un filtrado colaborativo basado en usuario, el vecindario se compondrá de un conjunto de usuarios ordenados por similitud, en orden decreciente, tomando como referencia el usuario al que se le quiere recomendar. En el caso de utilizar filtrado colaborativo basado en ítem, la estructura es la misma, teniendo ítems en vez de usuarios en el vecindario.

Dichas similitudes representan el factor de semejanza entre los usuarios o ítems. Existen diferentes métodos para medirlas, en este trabajo se han estudiado las siguientes similitudes:

* **Coseno**: Quizá sea la más famosa de todas, su función es medir el ángulo entre los vectores (ratings) para cada par de usuarios o ítems. Si ambos vectores apuntan al mismo lugar (son iguales) el valor de la similitud tendrá como valor 1, en el caso contrario que sean ortogonales el coseno se anula, devolviendo una similitud de 0.

****

* **Pearson**: Muy similar al coseno, con la diferencia que Pearson tiene en cuenta la puntuación media del usuario o ítem para suavizar las puntuaciones con valores extremos.

****

Se han implementado dos variantes de ésta similitud, ya que a la hora de realizar el cálculo se pueden tomar todos los usuario o ítems del dataset o simplemente aquellos que posean ratings en común.

* **Jaccard**: Mide el grado de similitud entre dos conjuntos como la cardinalidad de la intersección de ambos conjuntos dividida por la cardinalidad de su unión. Devuelve 0 si los conjuntos no poseen ningún valor en común, tendiendo a 1 a medida que aumenta el número de elementos compartidos.

****

***Factorización de matrices***

Existen otros métodos aparte no estudiados en este trabajo, como los algoritmos de factorización de matrices son bastante exitosos, caracterizan usuarios e ítems como vectores derivados de los patrones de ratings, dividiendo la matriz inicial en productos de diferentes matrices. Éstos métodos se han popularizado recientemente por tener un buen equilibrio entre escalabilidad y precisión en las recomendaciones, también son más flexibles a la hora de modelar situaciones del mundo real. Por norma general, los datos de entrada más convenientes para este tipo de algoritmos son ratings explícitos, pero en caso de no tenerlos, es posible añadir información adicional, utilizando ratings implícitos que reflejan indirectamente la opinión de los usuarios simplemente observando su comportamiento.

Hay diferentes variantes de algoritmos para obtener la factorización de matrices como pLSA (probabilistic Latent Semantic Analysis), SVD (Singular Value Decomposition), SVDN(SVD no-empty entries), HSVDN(SVD with Hypergraph transformation).

## 2.1.3 Algoritmos generales.

Los algoritmos generales no ofrecen recomendaciones personalizada, pues no tienen en cuenta ningún dato acerca de los usuarios.

***Popularidad***

Recomienda los ítems más populares, todos los usuarios tienen el mismo ranking. La popularidad implica el número de interacciones totales con un ítem, ya sean reproducciones de un vídeo o canción, productos más vendidos o los ratings más altos.

Éste algoritmo es de los que más se usan a nivel comercial, sin necesidad de utilizar un sistema de recomendación, debido a su fácil implementación y aporte de información útil para el usuario.

***Random***

Recomienda ítems de manera aleatoria, representa la probabilidad de escoger un ítem en conjunto de datos, por lo que cuanto más denso sea, más probabilidad de recomendar al usuario algo que le guste.

Comparación entre User kNN e Item kNN (4.2.5):

**Precisión**: depende del ratio entre el número de usuarios y los ítems del sistema, influyendo también si los ratings están distribuidos uniformemente sobre los items o no.

En general, es preferible un número pequeño de vecinos fiables a una gran cantidad de vecinos con una similitud no tan alta.

**Eficiencia**: tanto la memoria utilizada por un recomendador como el coste computacional dependen del ratio usuarios/items.

Por lo tanto, cuando el número de usuarios sobrepasa al número de ítems, correspondiendo al caso normal, la RBI requiere mucha menos memoria y coste, ya que se tienen que calcular muchas menos similitudes.

Aun así, el tiempo necesario para calcular una recomendación en la fase *online,* que depende solamente del nº de ítems relevantes y el nº máximo de vecinos es el mismo para los dos algoritmos.

En la práctica, el coste necesario para computar las similitudes es mucho menos costoso que el peor caso (teóricamente hablando) dado que los usuarios puntúan unos pocos ítems.

**Estabilidad**: la elección entre estos dos algoritmos se puede basar en la frecuencia y cantidad de cambios en los usuarios e ítems del sistema.

Si la lista de ítems es bastante estática en comparación a la de usuarios, una RBI es preferible dado que las similitudes entre ítems se pueden calcular en intervalos de tiempo separados, pudiendo de esta manera realizar recomendaciones a usuarios nuevos.

Por el contrario, en sistemas donde el número de ítems cambia constantemente, los métodos basados en usuarios prueban ser más estables.

Un aspecto importante que diferencia a estos algoritmos es por ejemplo la variedad,

mientras que una RBI tendrá poca variedad, pues siempre recomendará ítems parecidos que le hayan gustado al usuario, provocando cierto estancamiento, una RBU escogerá ítems algo más distantes a los gustos iniciales del usuario, que en ciertos casos es preferible al “estancamiento” antes mencionado.

**Normalización de ratings (4.3.1)**:

La normalización se aplica para acotar el valor de un rating a una escala deseada. Existen ciertas variantes:

* Mean-Centering: se compara el rating calculado con la media del usuario objetivo. Así se puede saber la apreciación (positiva o negativa) de un cierto usuario por un ítem observando el ítem normalizado.
* Z-score: supóngase el caso donde se tienen dos usuarios: A y B, el primero de ellos, posee una dispersión de ratings alta (varían mucho). Mientras que el segundo siempre puntúa con un rating igual a 3.

El hecho de que B puntúe un ítem con un rating de 5 implica más apreciación por parte de ese usuario que si dicha puntuación se la hubiera dado A. En mean-centering se eliminan estas desviaciones tomando únicamente la media.

Esta apreciación es la que mide z-score dividiendo el mean-centering del usuario entre la desviación típica de los ratings.

**Selección de vecinos (4.3.3.1)**:

El número de vecinos próximos y el criterio utilizado puede tener un gran impacto en la calidad del sistema de recomendación. La selección de los vecinos se lleva a cabo en dos pasos:

* Filtrado global de donde solamente los candidatos más afines son seleccionados.
* Una fase de predicción donde se escogen los posibles mejores candidatos.

Filtrado global de vecinos:

* Top-N: Para cada usuario, se guardan los N vecinos más cercanos con sus respectivas similitudes. Es necesario escoger de manera cautelosa este N por motivos de rendimiento y precisión. Si N es muy grande, se necesitará gran cantidad de memoria para almacenar todo el vecindario con los datos asociados y la predicción será lenta. Sin embargo, escogiendo un N muy pequeño es probable que algunos ítems nunca sean recomendados debido a la baja cobertura.
* Threshold: En vez de mantener un número fijo de vecinos (N), Threshold establece un umbral (w min) el cual deben superar los valores de las similitudes.

Esta técnica es algo más flexible que la anterior, pues solo los vecinos más significativos son escogidos, aun así dicho umbral es más complicado de establecer.

* Negative filtering: Por lo general, las correlaciones negativas entre usuarios son menos fiables que las positivas, esto se debe intuitivamente a que una correlación positiva entre dos usuarios es un buen indicador de su correspondencia a un grupo determinado (adolescente, fans de un género determinado…).

Aun así, una correlación negativa puede implicar pertenencia a grupos opuestos, y dependiendo del conjunto de datos en el que se esté trabajando, se pueden ignorar.

# Diseño

## Subsección

### Subsubsección

# Desarrollo

## Subsección

### Subsubsección

# Integración, pruebas y resultados

# Conclusiones y trabajo futuro

## Conclusiones

## Trabajo futuro

# Referencias

1. En las referencias figurarán los autores (opcionalmebte los editors), el título del artículo, el nombre de la revista o libro, el volumen y número de la revista, las páginas del artículo, la fecha de edición,. A continuación se listan algunos ejemplos
2. K.N. Platanioitis, C.S. Regazzoni (eds.), “Special Issue in Visual-centric Surveillance Networks and Services”, IEEE Signal Processing Magazine, 22(2), Marzo 2005.
3. B.S. Manjunath, P. Salembier, T. Sikora (eds.), “Introduction to MPEG 7: Multimedia Content Description Language,”, John Wiley and Sons, 2002
4. G. R. Bradski, “Computer vision face tracking as a component of a perceptual user interface,” en Proc.IEEE Workshop on Applications of Computer Vision, Princeton, NJ, October 1998, pp. 214–219.
5. A. D. Bue, D. Comaniciu, V. Ramesh, and C. Regazzoni, “Smart cameras with real-time video object generation,” in Proc. IEEE Intl. Conf. on Image Processing, Rochester, NY, volume III, 2002, pp. 429–432.
6. P. Anandan. “A computacional cuadrowork and an algorithm for the measurement of visual motion”, International Journal of Computer Vision, 2(3):283-310, January, 1989.
7. W.J. Ruckelidge. “Efficient Computation of the minimum Hausdorff Distance for Visual Recognition”, Phd thesis, Cornell Universitym 1995. CS-TR1454
8. “Extensible Markup Language (XML) 1.0 (Second Edition)”, W3C Recommendation 6 October 2000 <http://www.w3.org/TR/REC-xml>
9. William H. Press, Saul A.Teukolsky, William T. Vetterling, Brian P. Flannery. “Numerical Recipes in C – The art of Scientific Computing 2nd Edition”. Cambridge University Press

# Glosario

API Application Programming Interface

# Anexos

## Manual de instalación

## Manual del programador

## Anexo …

## 