Snake Table: A Dynamic Pivot Table for Streams of k-NN Searches

Juan Manuel Barrios*, **Benjamin Bustos***, Tomas Skopal^

* KDW+PRISMA, University of Chile

^ SIRET, Charles University in Prague

SISAP 2012, Toronto - Canada, August 2012

Motivation

- Video copy detection
- Observations
  - Consecutive queries are similar
  - Long query streams
  - Cheap distance function
- Is it possible to take advantage of the properties of query streams for improving the efficiency of k-NN?

Outline

- Streams of k-NN searches
- D-file and D-cache
- Snake Table and snake distribution
- Experimental evaluation
- Conclusions and future work

Streams of k-NN searches

- Sequence of queries
  - May have properties that can be exploited
- Example: queries from videos
  - Queries are frames (images) from the video
  - Usually 25 frames per second
  - Consecutive frames from the same shot are similar
    - Previous query could be used as an effective pivot!

Related work: D-file and D-cache

- D-file: just the original database using sequential scan, BUT
- it uses D-cache
  - a memory-resident structure that maintains the distances computed during previous queries
  - **provides lower-bounds (pivot based)** of requested distances that can be used to filter some of the database objects when querying
  - **O(1)** complexity for a lower bound retrieval
- **no preprocessing of database**

Related work: D-file and D-cache

- D-file works well if distance computation is "expensive"
- Otherwise, the overhead of D-cache may be too high, even if it discard many distance computations
  - Hash function computation
  - Distance insertion + replacement cost (collision resolution)

Snake Table

- Pivot-based index aimed to:
  - Improve the search time for streams of queries where consecutive query objects are similar
    - We call this “snake distribution”
  - Keep its internal complexity low to be applied in systems that use fast distance functions
    - E.g., CBVCD systems and interactive CBMIR that use global descriptors and Minkowski distances

Snake distribution

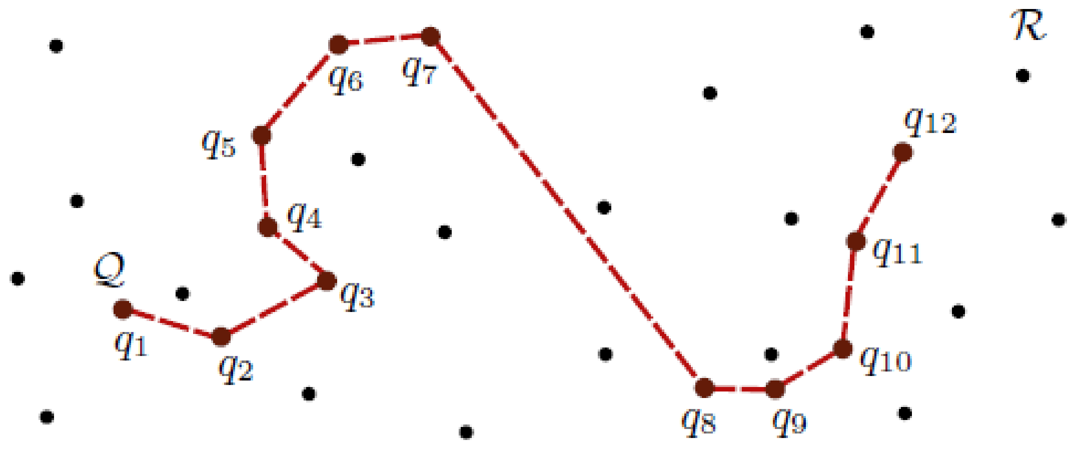


Fig. 1. Stream of queries $\mathcal{Q}=\{q_1, \ldots, q_{12}\}$ with a snake distribution: most of distances $d(q_i, q_{i+1})$ are smaller than $d(x, y)$ for randomly selected pairs x,y in $\mathcal{R}$.

Snake Table

- Life cycle
  - When a new session starts, an empty Snake Table is created
  - When a query q is received:
    - k-NN is performed
    - Distances computed are stored in the table
    - Result is returned
  - In the following queries
    - Previous query objects are used as pivots
  - When the session ends, table is discarded

Snake Table

- Data structure
  - Fixed-sized matrix used as a dynamic pivot table (p pivots)
  - Each cell in the j-th row contains a pair $(q,d(q,o_j))$ for some q (not necessarily in order)
  - At query time
    - Lower bound distance is computed for discarding o_j
    - If object o_j is not discarded, computed distance is stored in the table

Snake Table

- Replacement strategies
  - V1: round-robin mode
    - If distance was not computed
      - Cell is left unmodified, but must be checked in further queries before computing lower bound
  - V2: highest distance in the row is replaced
  - V3: "independent" round-robin
    - for each row, every rows compactly stores the last p evaluated distances
    - Lower bound distance computed from last query and goes backwards

Experimental evaluation

- Dataset
  - MUSCLE-VCD-2007 (Video copy database)
  - Descriptors:
    - Edge Histogram
    - Ordinal Histogram
    - Color Histogram
    - Keyframe
    - Linear combinations of these descriptors
  - Distance: L1 (Manhattan)

Experimental evaluation

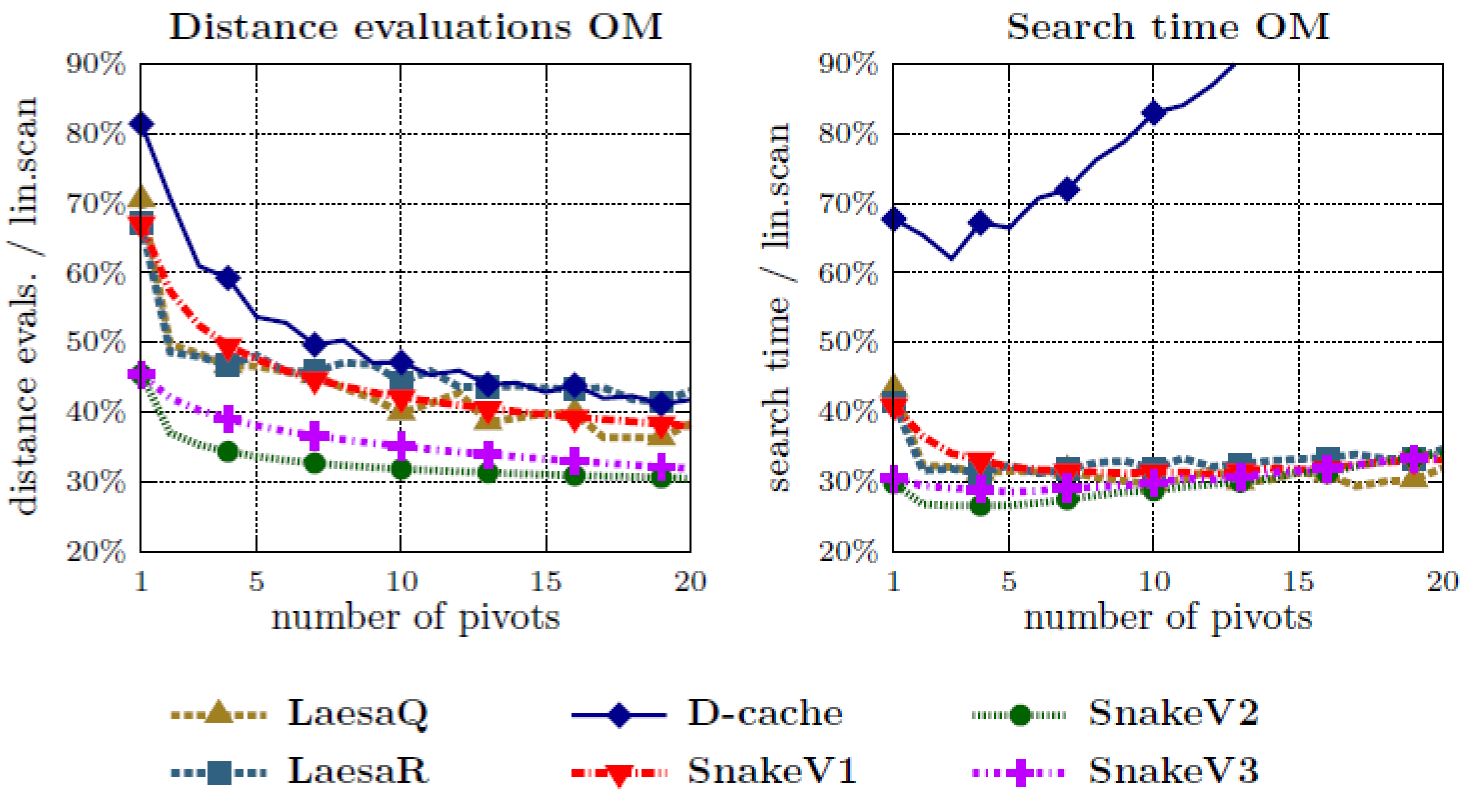
- Indexes
  - D-cache
  - LAESA
    - LAESA-R: choose pivots from data set
    - LAESA-Q: choose pivots from queries
    - Pivots chosen using SSS (Sparse Spatial Selection)
  - Snake Table: SnakeV1, SnakeV2, SnakeV3
- All indexes of same size
- p varies between 1 and 20

Experimental evaluation

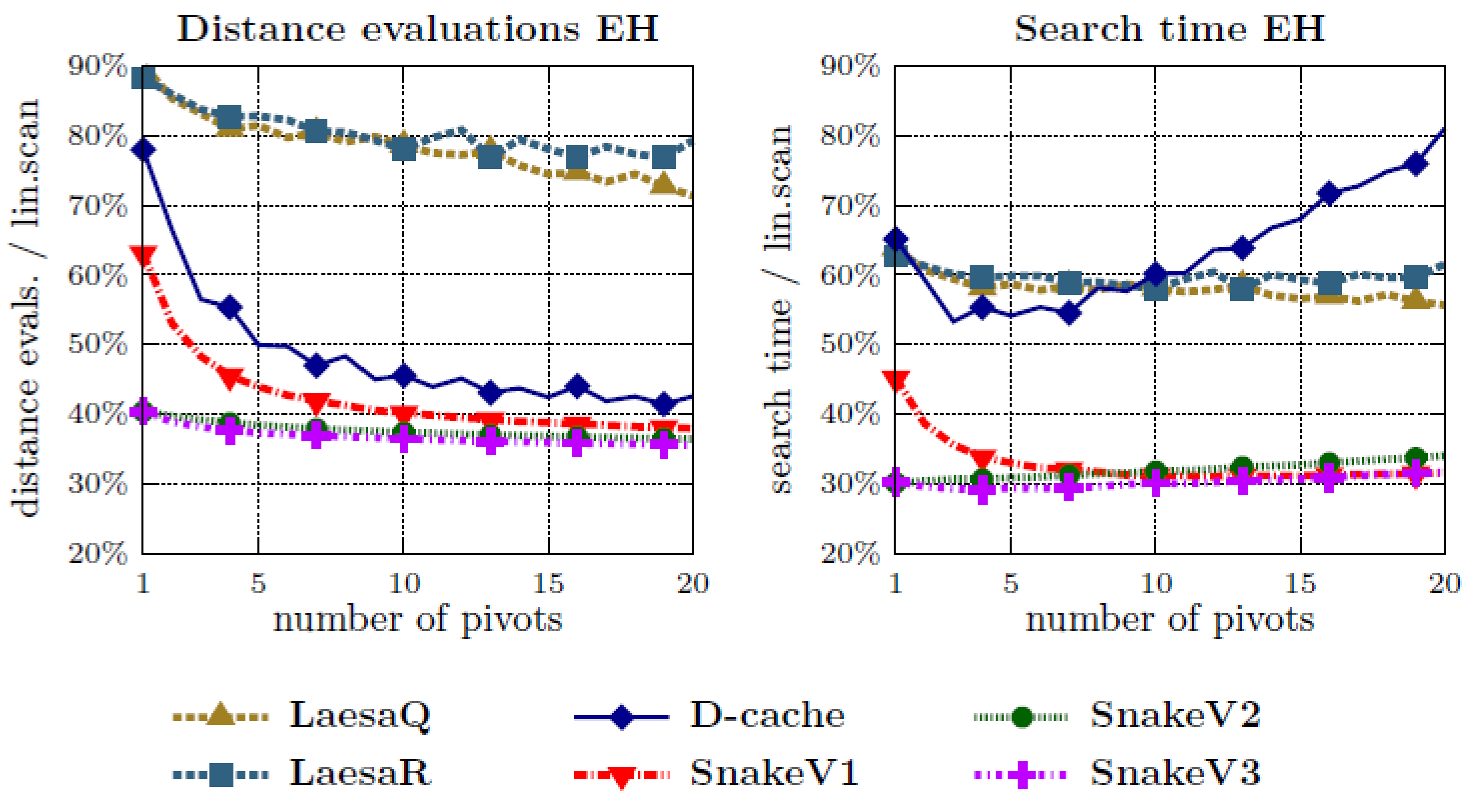
	Time	MAP	max	μ	σ	ρ	$\mathbf{H}_d$
Group 1							
OM	282 s.	0.125	3285	1489	416	6.4	
KF	304 s.	0.509	24721	7264	2636	3.8	
Group 2							
EH	541 s.	0.639	7996	3198	751	9.1	
CH	501 s.	0.482	6219	3661	970	7.1	
Group 3							
ECK	1258 s.	0.646	0.888	0.416	0.09	11.4	
EK3	2214 s.	0.732	0.870	0.347	0.08	10.2	

Table 1. Effectiveness and efficiency for the base configurations.

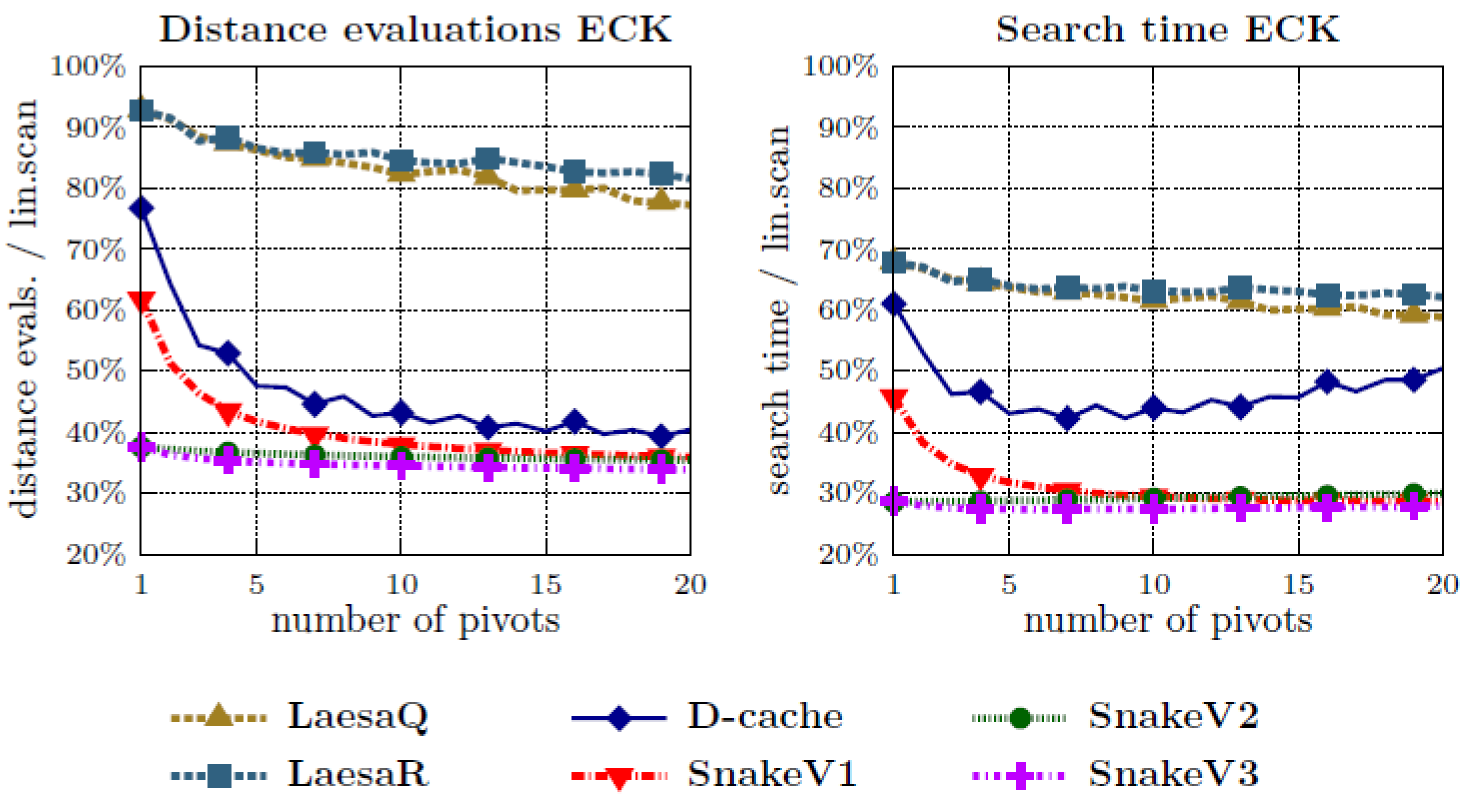
Experimental evaluation



Experimental evaluation



Experimental evaluation



Conclusions and future work

- Snake Table achieves high performance with queries that follows a snake distribution
  - This is due to dynamic selection of good pivots
  - It's better to avoid empty or unused cells
- No preprocessing needed
- Better alternative than D-cache in the tested scenarios

Conclusions and future work

- It requires space proportional to the dataset
  - Not memory efficient
- Suitable for medium-sized data sets with long k-NN streams (like in video retrieval)

Conclusions and future work

- Future work:
  - When p is high, many pivots are close to each other
    - They may become redundant
    - Possible solution: use a mix of static and dynamic pivots
  - Solve parallel queries with Snake Table

Thank you for your attention!

PRISMA

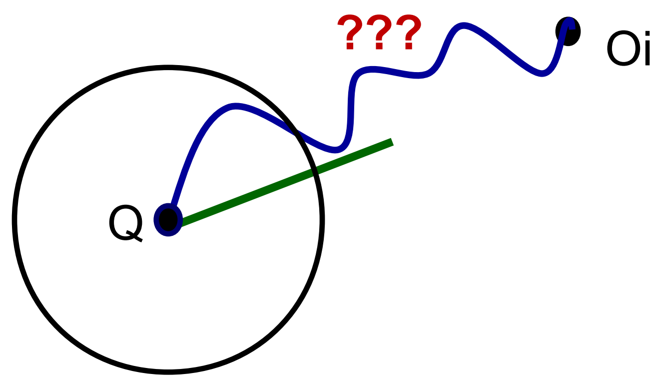
Pattern Recognition, Indexing and Similarity search in Multimedia Archives

siret

research group

This slide has been intentionally left blank

D-file – range query



set **RangeQuery**(Q, r_Q) {

for each O_i **in** database **do**

compute $\delta(Q, O_i)$;

if $\delta(Q, O_i) \leq r_Q$ **then** add O_i to the query result } // basic filtering

simple sequential search

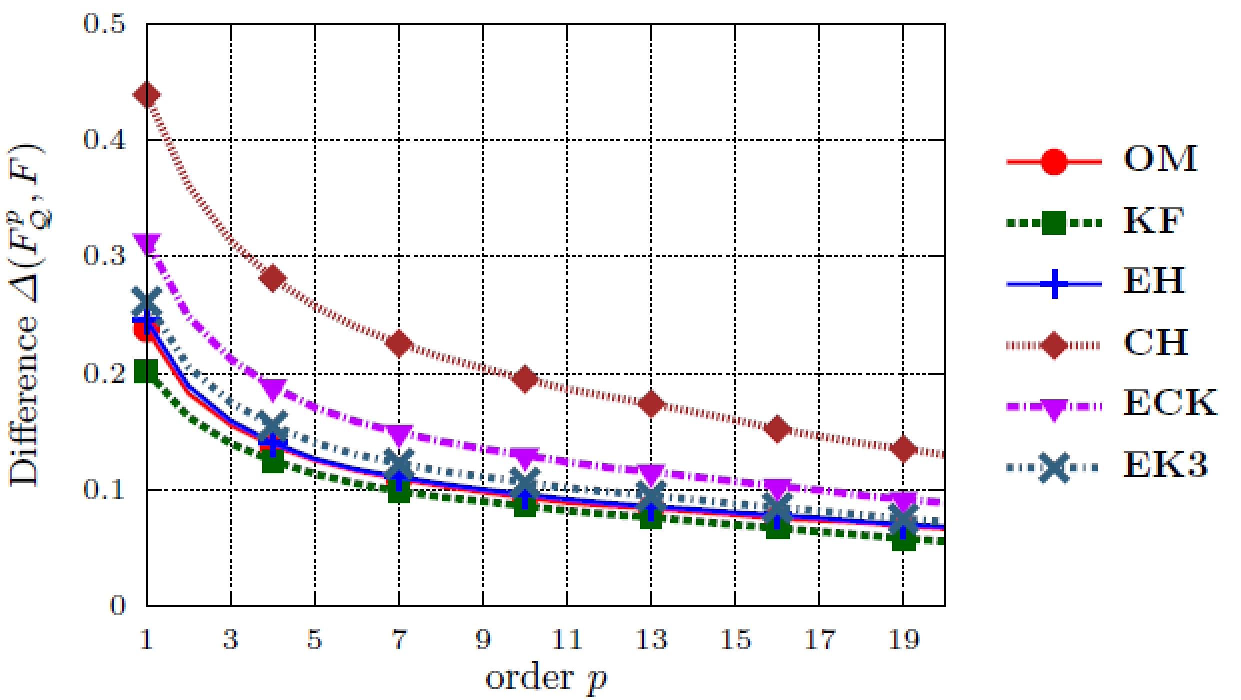
sequential search enhanced by D-cache filtering

Snake distribution

- Formal definition:

Definition 3 *(Snake Distribution) Let* $\mathcal{M} = (\mathcal{D}, d)$ *be a metric space, let* $\mathcal{R} \subset \mathcal{D}$ *be the collection of objects, and let* $\mathcal{Q} \subset \mathcal{D}$ *be a set of* m *query objects* $\mathcal{Q} = \{q_1, ..., q_m\}$*. Let* F *be the cumulative distribution of* $d(x, y)$ *with random pairs* $x, y \in \mathcal{Q} \cup \mathcal{R}$*,* p *be a number between 1 and m-1, and* $F_{\mathcal{Q}}^{p}$ *be the cumulative distribution of* $d(q_i, q_{i-p})$ $\forall\, i \in \{p+1, ..., m\}$*.* $\mathcal{Q}$ *fits a snake distribution of order* p *if* $\Delta(F_{\mathcal{Q}}^{p}, F) > s$*, for some threshold value* $s \in \mathbb{R}^+$*.*

Experimental evaluation



Experimental evaluation

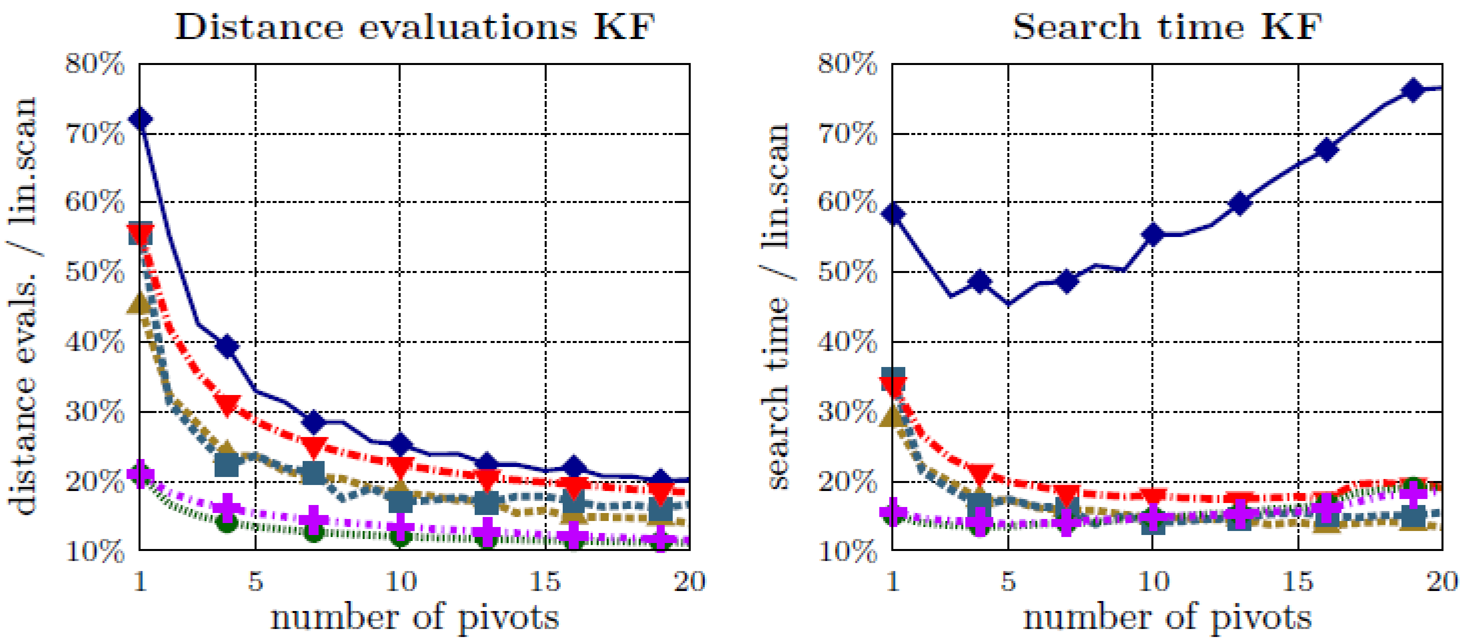


Fig. 3. Search time and distance evaluations for **OM** and **KF** (Group 1).

Experimental evaluation

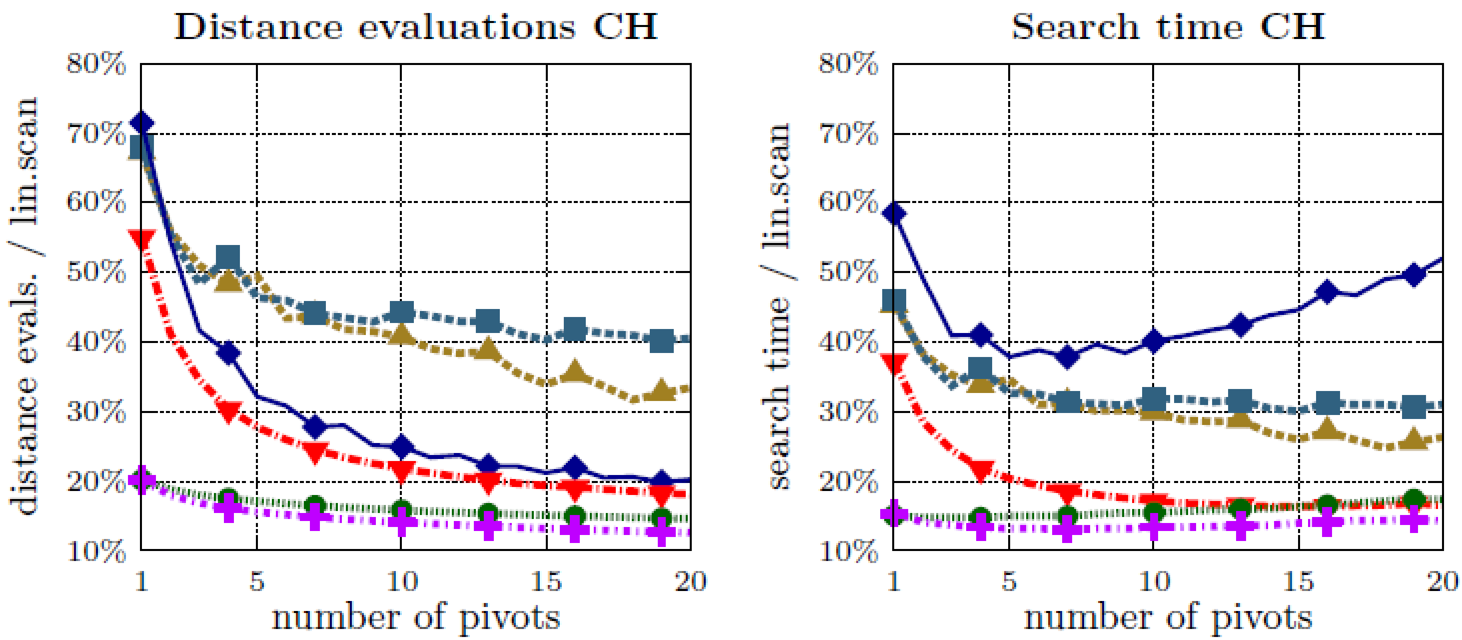


Fig. 4. Search time and distance evaluations for **EH** and **CH** (Group 2).

Experimental evaluation

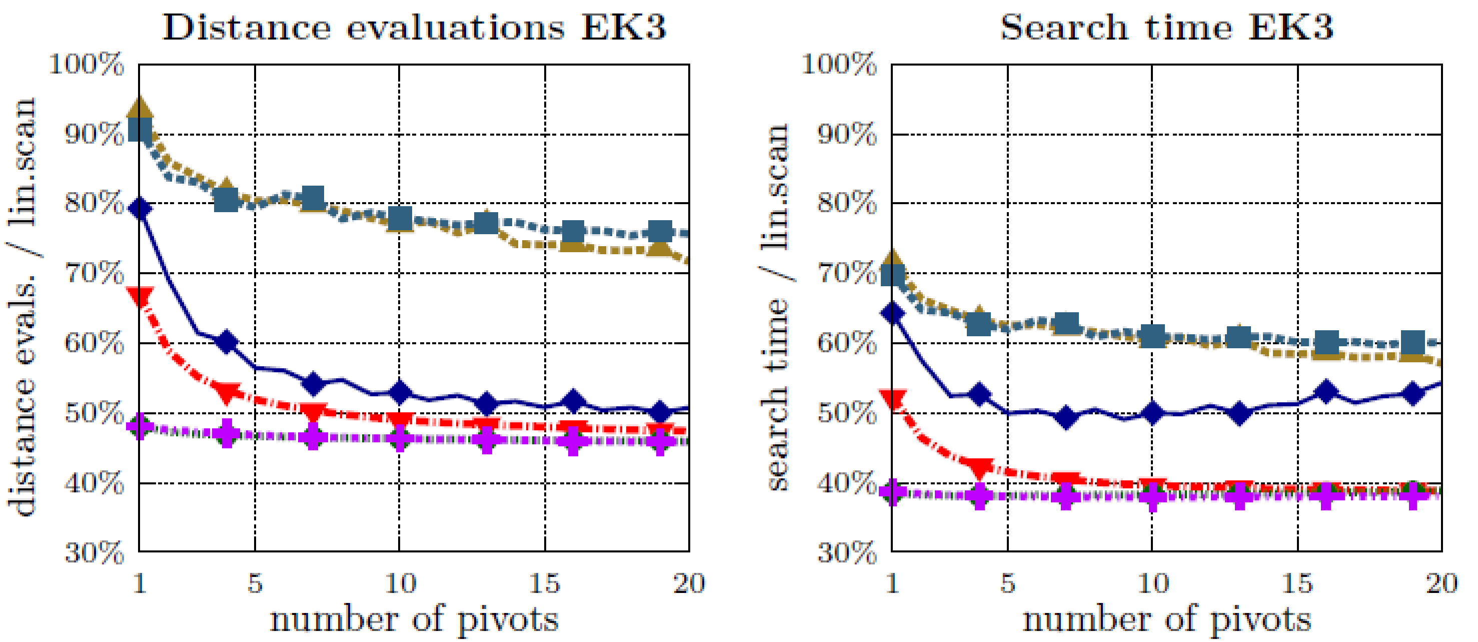
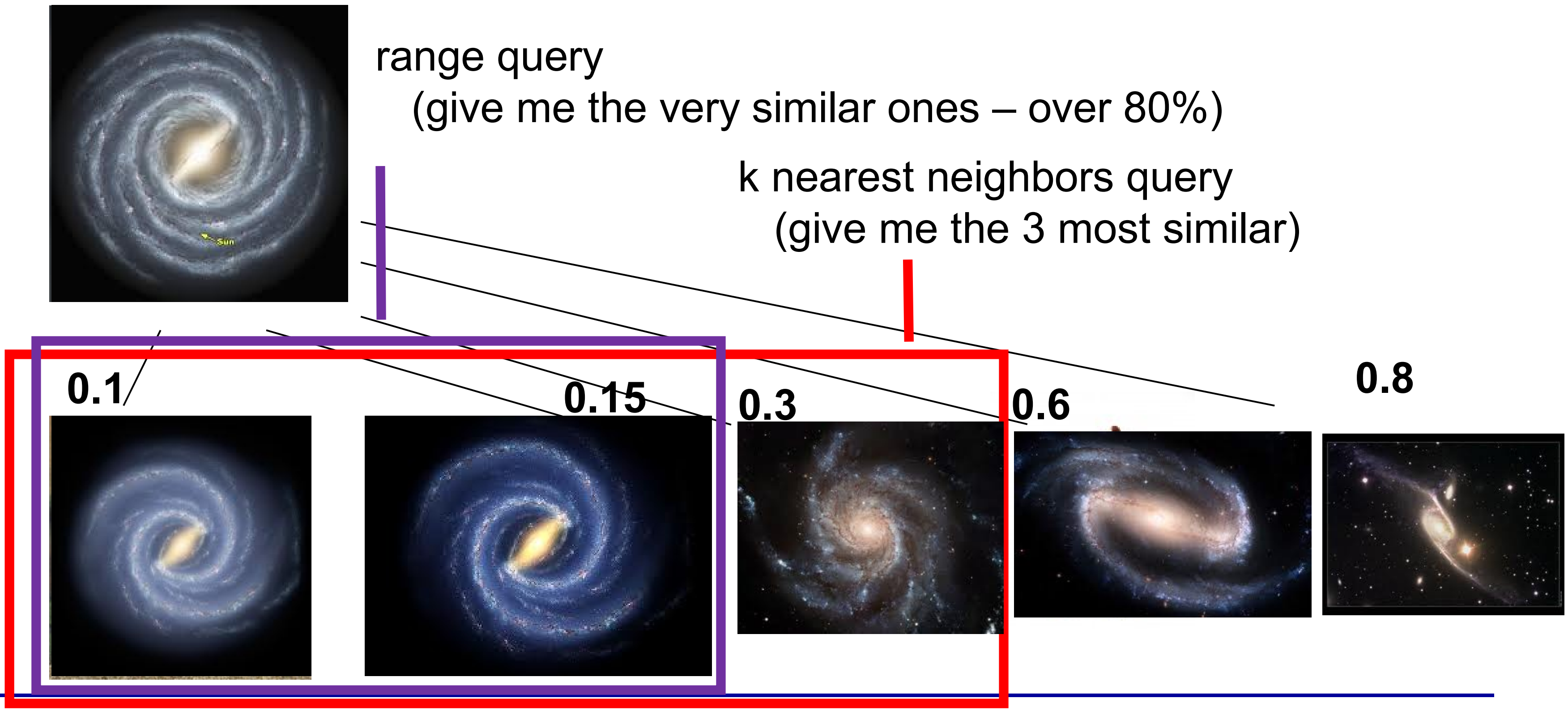


Fig. 5. Search time and distance evaluations for **ECK** and **EK3** (Group 3).

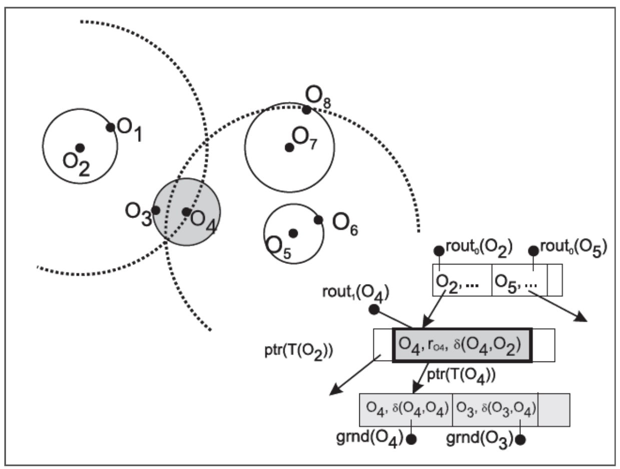
Similarity search

- Multimedia databases, time series, bioinformatics, ...
- Content-based similarity search (query by example)

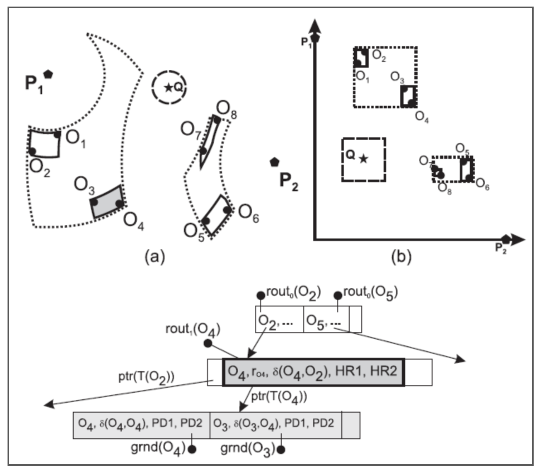


Index-based metric access methods

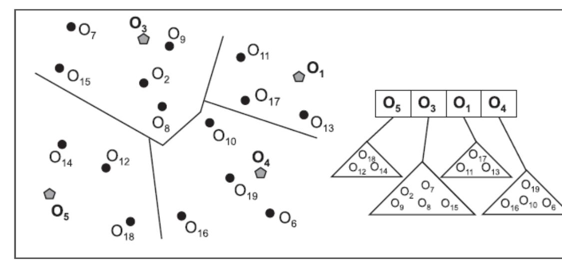
- All metric access methods (MAM) are **index-based**, i.e., preprocessing of a database is always needed.
- Index construction takes between O(**n** log **n**) and O(**n**²).



M-tree



PM-tree



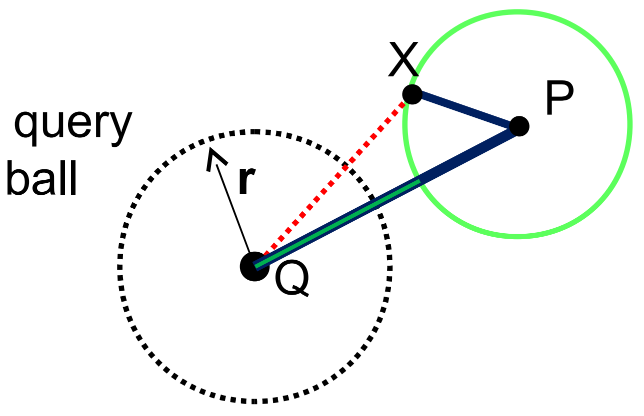
GNAT

Outline

- Pivot-based indexing
- Motivation for index-free similarity search
- D-file (+ D-cache)
- Snake Table
- Final remarks

Using lower-bound distances for filtering database objects

- cheap determination of **lower-bound distance** of **δ(*,*)**



<u>The task:</u> check if **X** is inside query ball
•we know **δ(Q,P)**
•we know **δ(P,X)**
•we do not know **δ(Q,X)**
•we do not have to compute **δ(Q,X)**, because its lower bound **|δ(Q,P)-δ (X,P)|** is larger than **r**, so **X** surely cannot be in the query ball, so **X** is ignored

- this filtering is used in various forms by metric access methods, where **X** stands for a database object and **P** for a pivot object

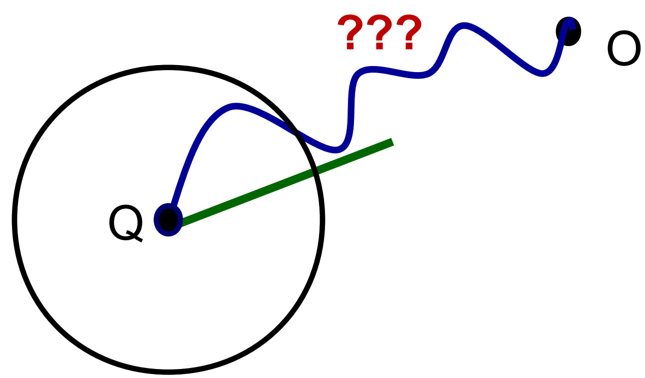
Motivation for index-free search

- indexing is not desirable (or even possible) if
  - we have a highly **changeable** database
    - more inserts/deletes/updates than searches, i.e., streaming databases, archives, logs, sensory databases, etc.
  - we perform **isolated** searches
    - a database is created for a few queries and then discarded, i.e., in data mining tasks
  - we switch between distances (**changing similarity**)
    - the distance function is tuned at query time, e.g., weighing of object features is applied dynamically

D-file

- just the original database using sequential scan, BUT
- it uses D-cache
  - a memory-resident structure that maintains the distances computed during previous queries
  - **provides lower-bounds** of requested distances that can be used to filter some of the database objects when querying
  - **O(1)** complexity for a lower bound retrieval
- **no preprocessing of database**

D-file – range query



set **RangeQuery**(Q, r_Q) {

for each O_i **in** database **do**

compute $\delta(Q, O_i)$;

if $\delta(Q, O_i) \leq r_Q$ **then** add O_i to the query result } // basic filtering

simple sequential search

sequential search enhanced by D-cache filtering

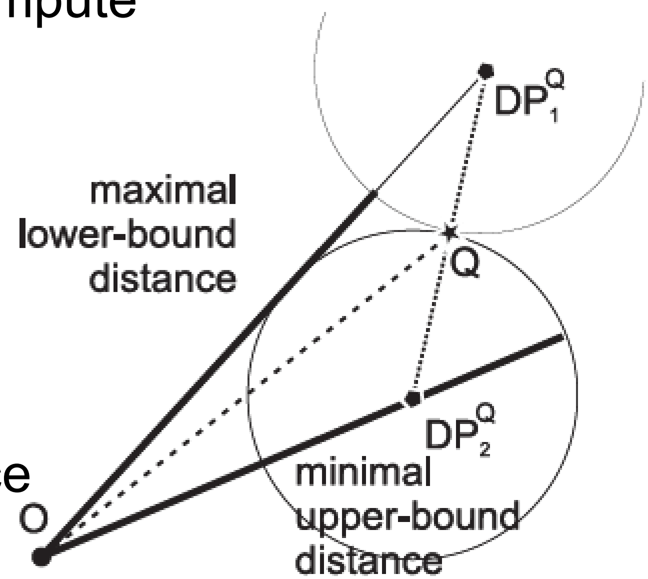
D-cache

- every time a D-file computes a distance δ(*,*), it is stored into D-cache
- the D-cache could be viewed as a sparse matrix, where queries denote row, database object denote columns, and a cell contains value of δ(Q,O)

$$D = \begin{array}{c} \\ Q_1 \\ Q_2 \\ Q_3 \\ \dots \\ Q_m \end{array} \begin{array}{c} \begin{array}{cccccc} O_1 & O_2 & O_3 & \dots & O_n \end{array} \\ \begin{pmatrix} & d_{12} & d_{13} & \dots & \\ d_{21} & & & \dots & d_{2n} \\ & & & \dots & \\ \dots & \dots & \dots & \dots & \dots \\ d_{m1} & & d_{m3} & \dots & \end{pmatrix} \end{array}$$

D-cache

- D-cache has two functionalities
  - it allows to retrieve the exact distance $\delta(Q,O)$, if it is there
  - the main functionality: it provides *tight lower bound* to $\delta(Q,O)$
- How to obtain a lower bound?
  - prior to a new query Q, determine some old queries DP_i^Q (acting as **dynamic pivots**) and compute the distances $\delta(Q, DP_i^Q)$
  - when a lower bound to d(Q,O) is required, search for available distances $\delta(Q, DP_i^Q)$ in the D-cache and obtain the *max*$(|\delta(DP_i^Q, O) - \delta(Q, DP_i^Q)|)$; that is our tight lower bound distance



D-cache

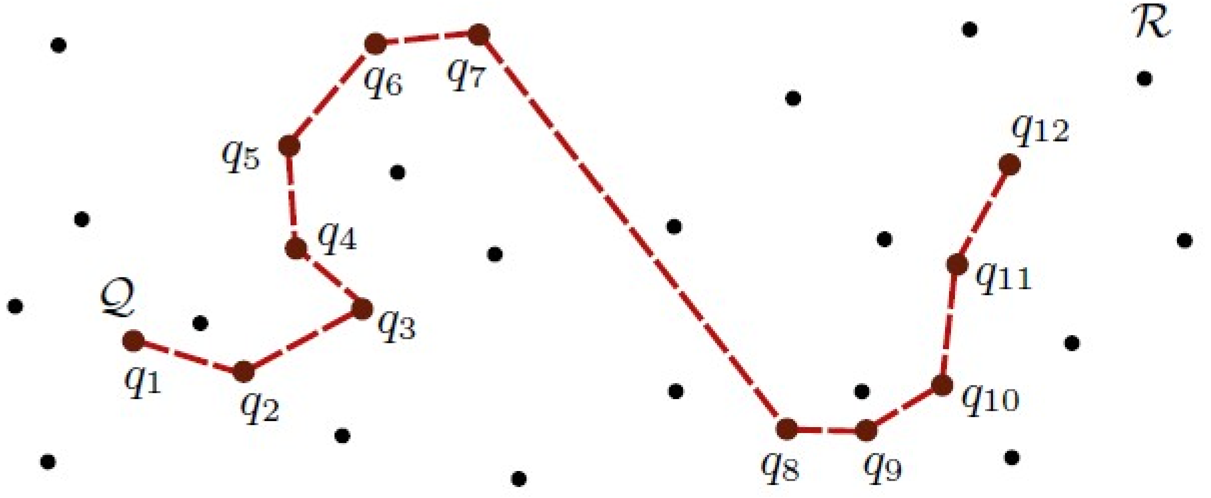
- How to choose the dynamic pivots?
  - "Recent" policy
    - simple – we just choose *k* previous queries
    - motivation: the recently added distances are likely to still sit in the D-cache
- Data structure: hash table
  - determine individual cell values based on id_1, id_2
  - "Simple" or universal hashing
- Distance insertion
  - Each computed distance is inserted into D-cache
  - Replacement policies
    - obsolete distances (from outdated pivots)
    - distance-based

Snake Table

- D-file works well if
  - distance function is "expensive"
  - problem: overhead (hashing, replacement policy, etc.) is not negligible for "cheap" distances
    - it may avoid many distance computations but the total search time will be large
- Snake Table
  - designed for streams of k-NN queries
  - no preprocessing required
  - query objects fits a "snake distribution"

Snake Table

- "Snake distribution"
  - consecutive queries are close
  - e.g.: frames from a video shot



Snake Table

- Data structure
  - Table of size n*k
    - n: size of the data set
    - k: number of dynamic pivots
  - Dynamic pivots are replaced in round-robin mode
    - each query is a pivot for the next k queries
    - snake distribution: dynamic pivots are close to next query
- In practice: it performs better than D-file for “cheap” distances

Final remarks

- D-file – an index-free metric access method
  - requires no indexing
  - suitable for online streaming data processing
  - D-cache: a structure used by D-file to cheaply determine lower-bound distances
    - uses distances computed and cached during previous queries processing
- Snake Table
  - lower internal complexity compared with D-cache
  - faster than D-cache when data fit a “snake distribution”

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE CHILE

Thank you for your attention!

PRISMA
Pattern Recognition, Indexing and Similarity search in Multimedia Archives

This slide has been intentionally left blank

Datos multimedia

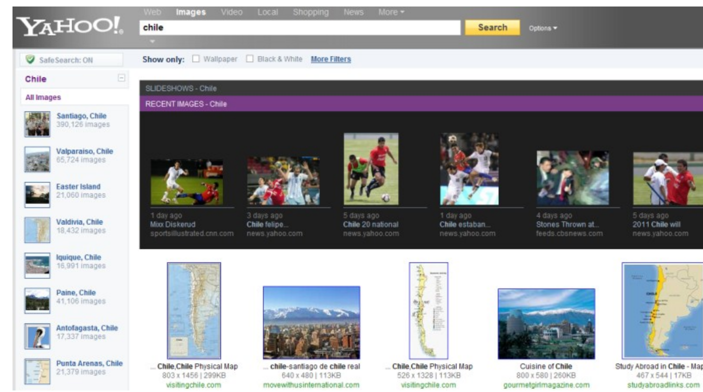
- Más del 95% del contenido Web son datos multimedia
  - Imágenes, Video, Audio
  - Cualquier dato digitalizado sin estructura
- Tendencia irreversible
  - Aparatos de captura de bajo costo
  - Internet de alta velocidad
  - Actividad humana en Internet (redes sociales e industria)

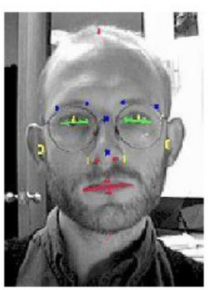
Recuperación de información multimedia

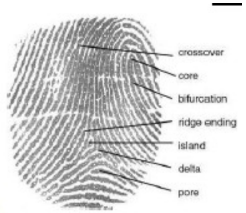
- Problemas principales
  - Búsqueda
  - Recuperación
- Problemas relacionados
  - Administración de contenido multimedia
  - Interacción con el usuario
  - Redes sociales

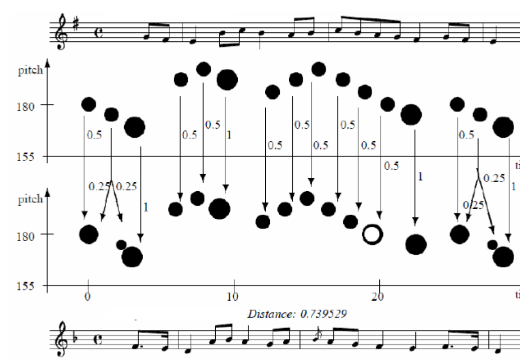
Recuperación de información multimedia

- Áreas de aplicación
  - Bases de datos científicas
  - Biometría
  - Reconocimiento de patrones
  - Industria manufacturera
  - Etc.









Búsqueda por similitud

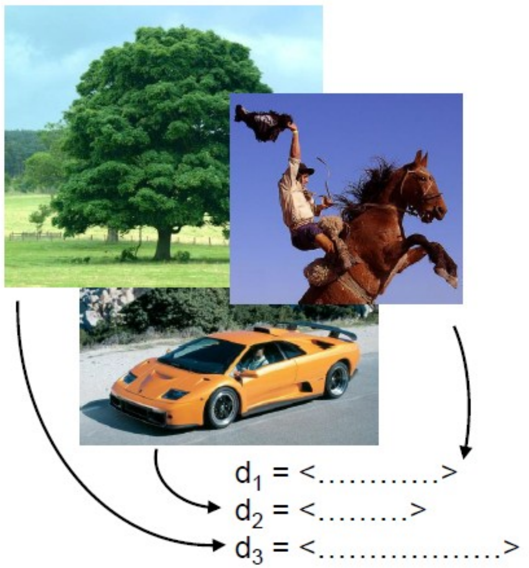
- Problema: encontrar objetos "parecidos" o "relevantes"
- Contexto vs. contenido
  - Contenido
  - Anotaciones manuales
  - Anotaciones automáticas

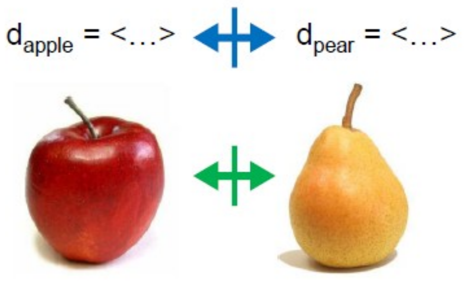
Búsqueda por similitud

- Búsqueda textual: buscadores Web
- Ventajas
  - Permite consultas semánticas de alto nivel
  - Fácil de implementar
- Desventajas
  - Requiere intervención humana
  - Altamente subjetivo
  - Incompleto

Búsqueda por similitud basada en contenido

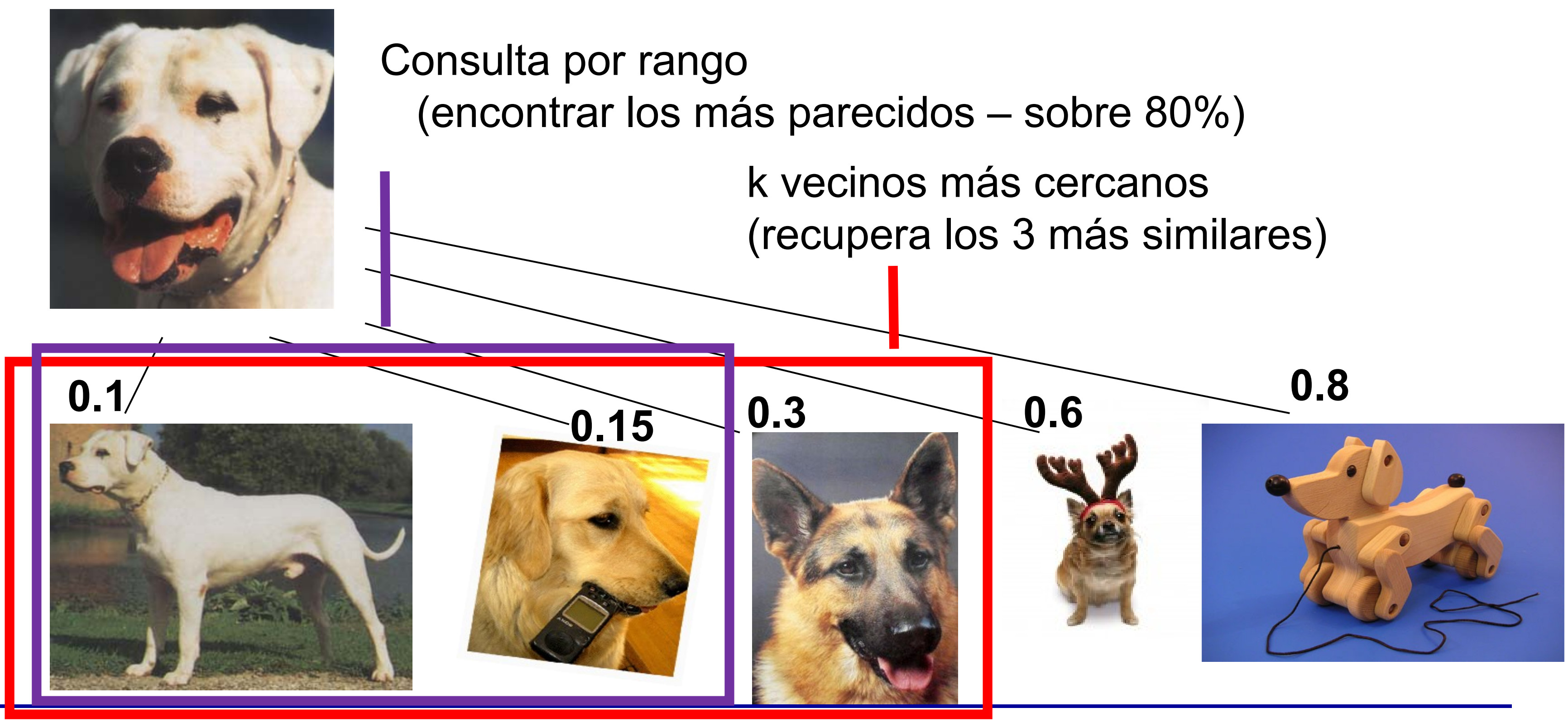
- Modelo de búsqueda
  - Extracción de características
    - Descriptor (vector)
    - Estructura del descriptor está oculta al usuario
  - Función de similitud
    - Permite comparar descriptores
    - Debe “imitar” la similitud semántica de los objetos





Búsqueda por similitud basada en contenido

- Tipos de consulta
  - Query-by-example



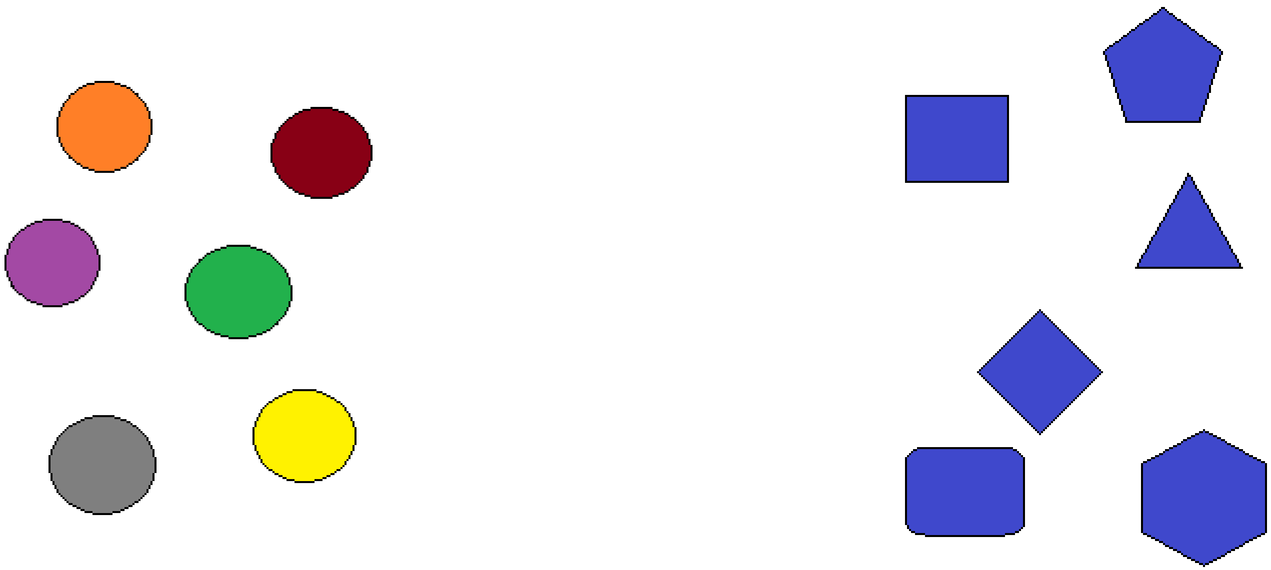
Búsqueda por similitud basada en contenido

- Espacios métricos
  - Función de disimilitud δ (distancia), universo **U**, colección **S** ⊂ **U**, objetos x,y,z ∈ **U**
  - A mayor δ(x,y), más disímiles son los objetos x,y
- Propiedades topológicas

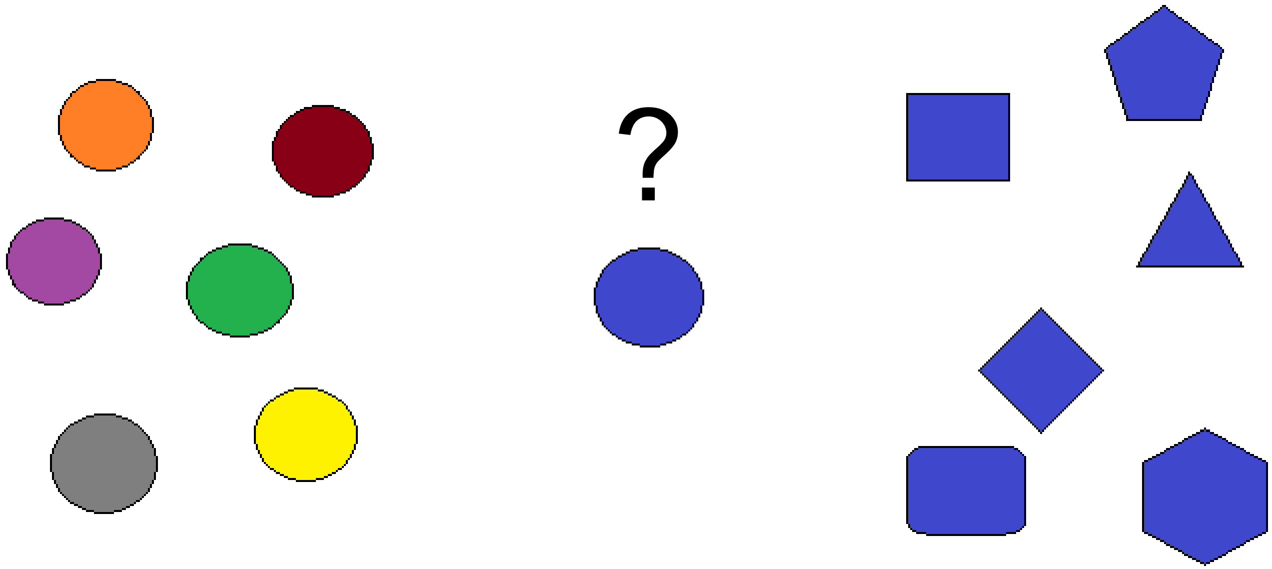
$$
\begin{array}{rr}
\delta(x,y) = 0 \Leftrightarrow x = y & \text{identity} \\
\delta(x,y) \geq 0 & \text{non-negativity} \\
\delta(x,y) = \delta(y,x) & \text{symmetry} \\
\delta(x,y) + \delta(y,z) \geq \delta(x,z) & \text{triangle inequality}
\end{array}
$$

- Ventajas de los espacios métricos
  - Se conocen muchas métricas
  - Postulados apoyan supuestos comunes sobre similitud
  - Permite indexamiento y búsqueda eficiente

Búsqueda por similitud basada en contenido

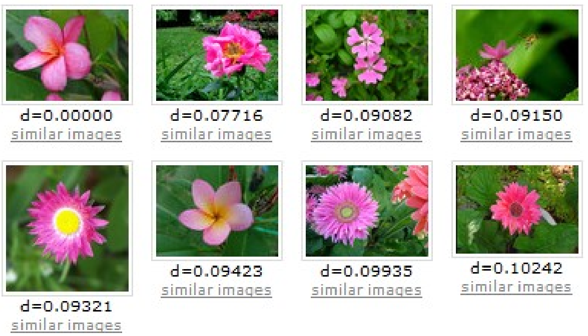


Búsqueda por similitud basada en contenido



Ejemplo: Búsqueda de imágenes

- Problema: encontrar imágenes parecidas

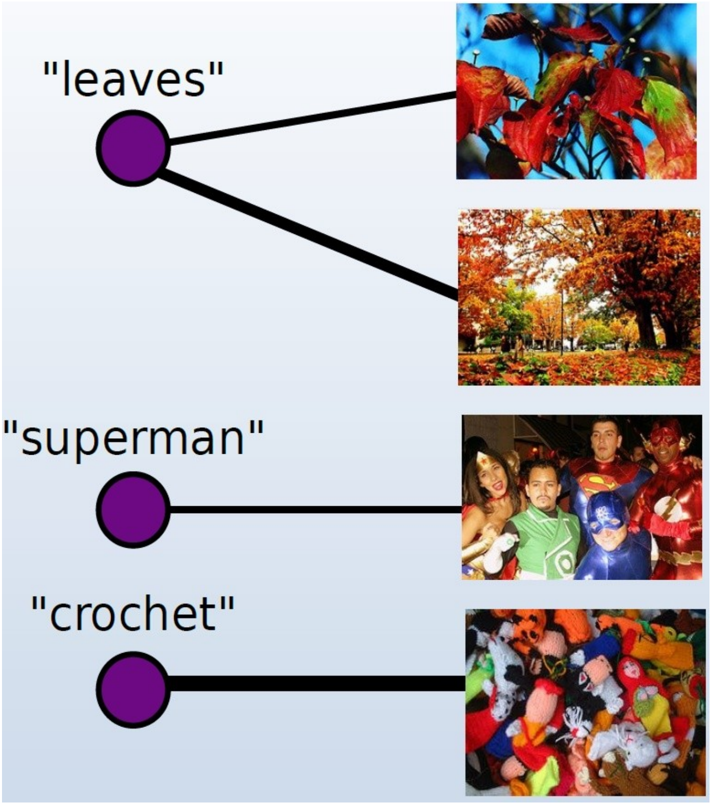


PRISMA Image Search:
http://prisma.dcc.uchile.cl/ImageSearch/

- Consulta: Texto, imagen, sketch, combinación

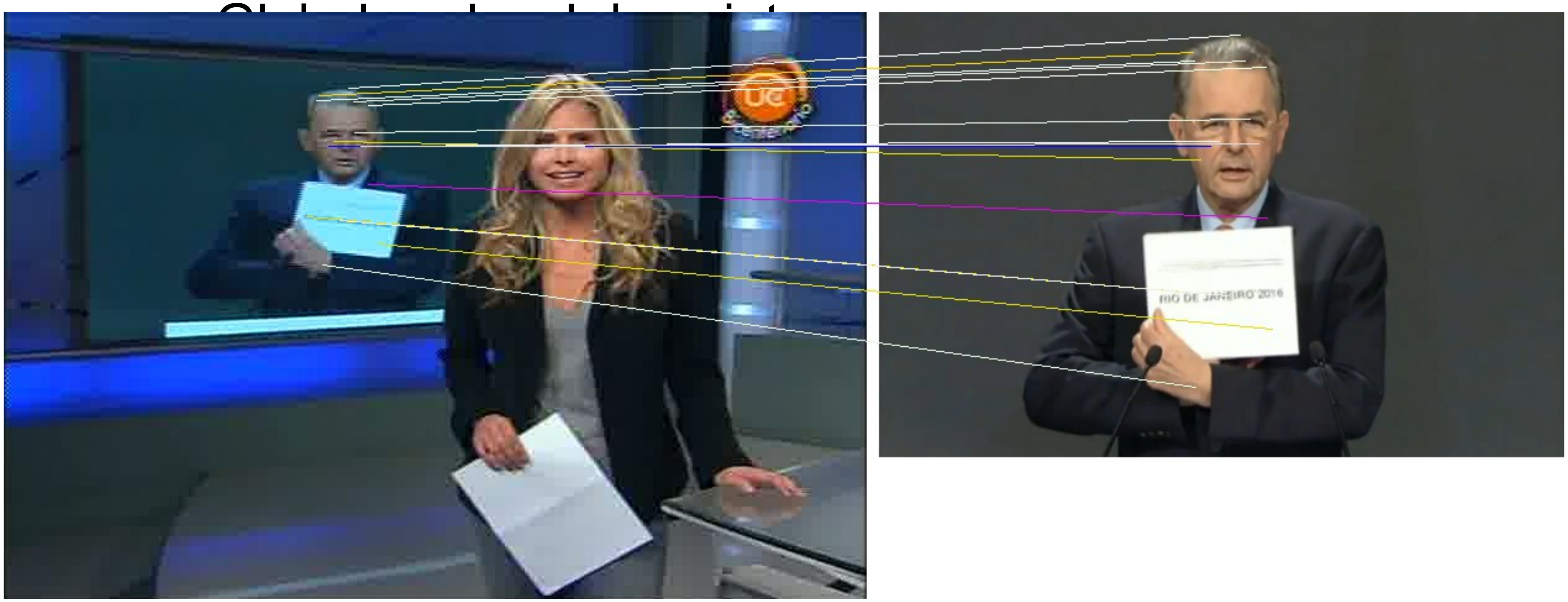
Ejemplo: Búsqueda de imágenes

- Descriptores para imágenes
  - Alto nivel: conceptos
    - Metadatos
      - Título, tags, etc.
    - Generados por usuario
      - Clic-logs
      - Contiene información semántica



Ejemplo: Búsqueda de imágenes

- Descriptores para imágenes
  - Bajo nivel: atributos visuales
    - Color, textura, forma, bordes

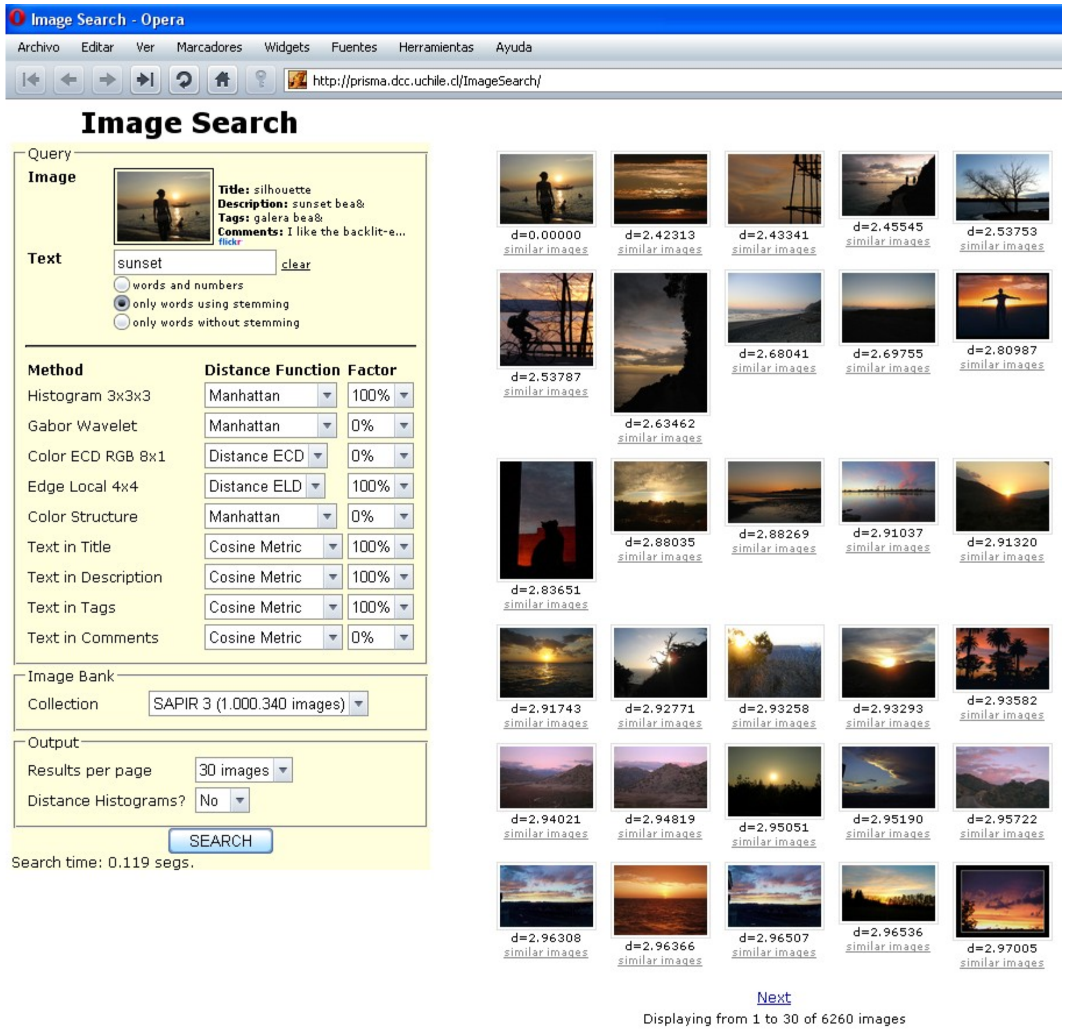


Ejemplo: Búsqueda de imágenes

- Gran problema: gap semántico
  - Brecha entre descriptores de alto y bajo nivel

(crédito: Google)

Buscador de imágenes PRISMA



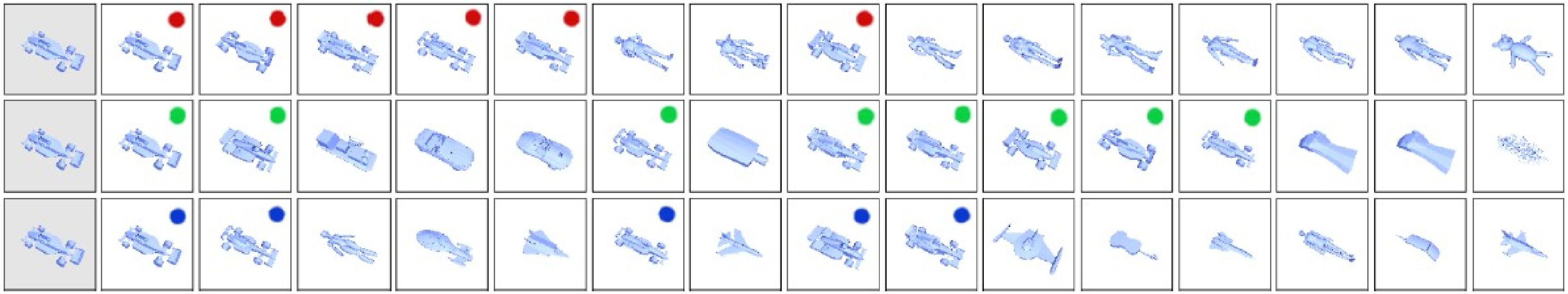
Temas de investigación

- Detección de copia en video
- Tagging automático de imágenes
- Indexamiento
- Búsqueda basada en sketchs
- Análisis de series temporales
- Búsqueda en modelos 3D
- Análisis formal de técnicas de indexamiento
- Búsquedas basadas en contenido y contexto

Experiencia de Transferencia Tecnológica: Chequemático

Motivación

- Proyecto de transferencia tecnológica: Búsqueda en colecciones CAD para la industria automotriz



Contacto inicial

- Mauricio Palma, Gerente General de Orand
  - Proyectos de ingeniería de software
  - Interesados en realizar proyectos de innovación
- Primera discusión
  - Presentación empresa y grupo de investigación
  - Intercambio de problemas – soluciones

Contacto inicial

- “Chequemático”
  - Depósito de cheques
  - Pago de cheques
  - Automatizado

Contacto inicial

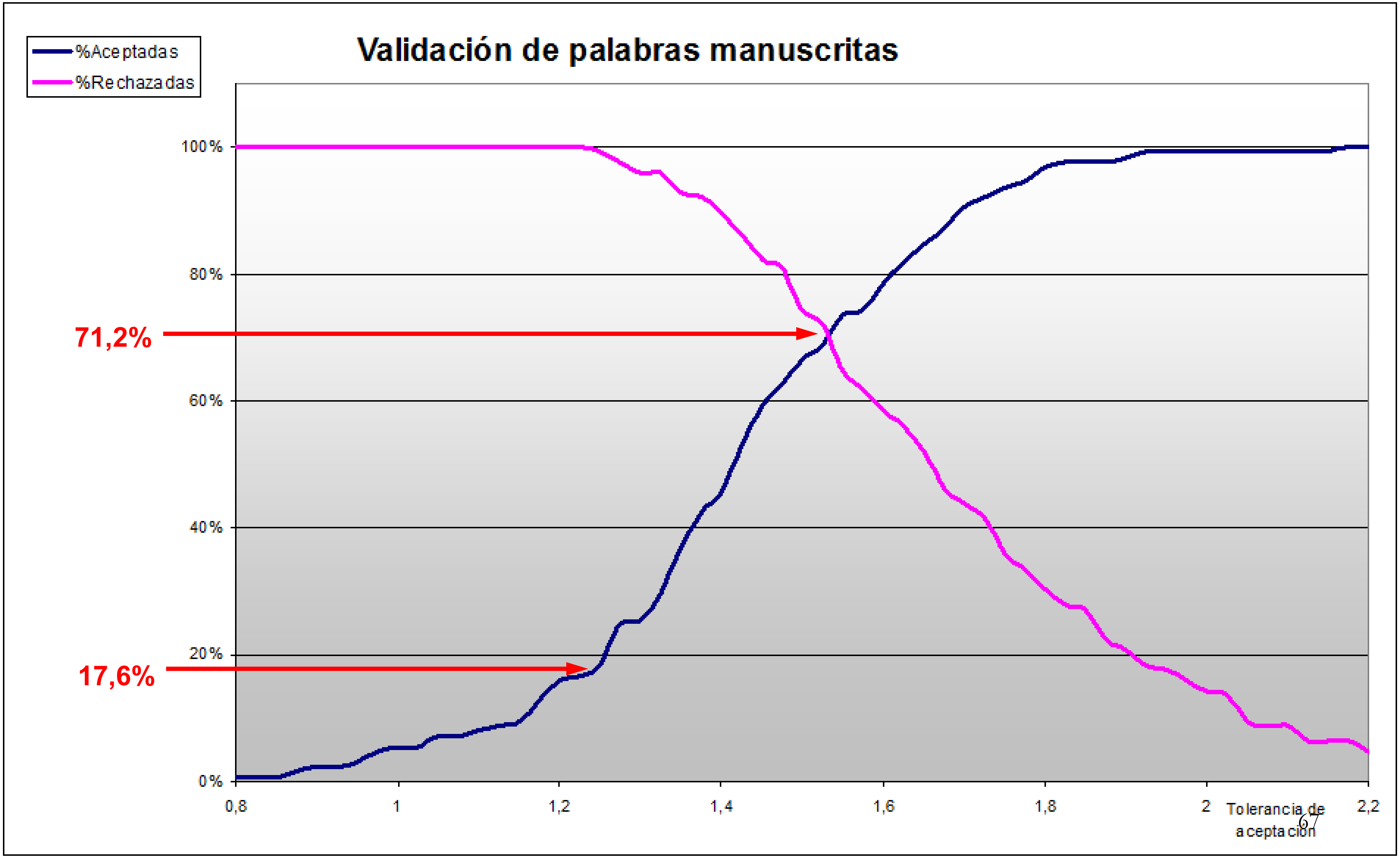
- El problema: verificación de nombre en cheque

  - Letra imprenta
  - Manuscritos
  - Sin/con ruido
  - Alineación

Desarrollo del proyecto

- Etapa I: estudio de factibilidad
  - Revisar el estado del arte (leer *papers*)
  - Implementar piloto inicial
  - Evaluación preliminar
    - *False accept rate* (FAR)
    - *False reject rate* (FRR)
    - *Equal error rate* (ERR)
  - En paralelo: Capacitación al personal de Orand

Desarrollo del proyecto



Desarrollo del proyecto

- Etapa II: *fine tuning* de los algoritmos
  - Revisión de los algoritmos, parámetros, etc.
  - Implementación de prototipos
  - Pruebas masivas
  - En paralelo: implantación de la tecnología (Orand)
- Segundo proyecto: verificación de endoso
  - Identificar firma
  - Identificar R.U.T. y número de cuenta corriente

Desarrollo del proyecto

Fases	DCC	Orand	BCI
Charlas de nivelación	✓		
Generación de datos de prueba (imágenes)		✓	✓
Desarrollo de métodos candidatos	✓		
Evaluación y selección de mejor método	✓	✓	
Desarrollo de métodos de pre-procesamiento		✓	
Pruebas masivas		✓	✓
Implementación en lenguaje de programación del cliente y mejoras de performance		✓	
Fases	**DCC**	**Orand** ✓	**BCI** ✓

Reflexiones

- Hay muchos problemas interesantes para resolver en el área *Multimedia – Pattern Recognition*
- Vital: entidad mediadora entre Universidad – empresa privada
  - Parte ejecutora
  - Implantación de la tecnología
- Universidad provee conocimiento de punta
- Centros de I+D en empresas privadas

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE CHILE

¡Gracias por su atención!

PRISMA
Pattern Recognition, Indexing and Similarity search in Multimedia Archives