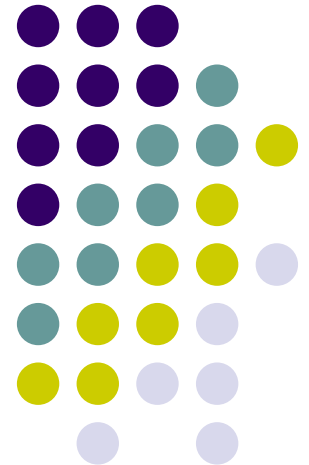


# Similarity Search: A Matching Based Approach

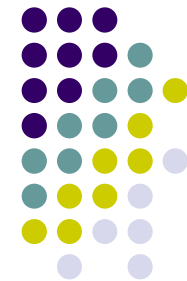
---

*Rui Zhang*

The University of Melbourne  
July 2006



# Outline

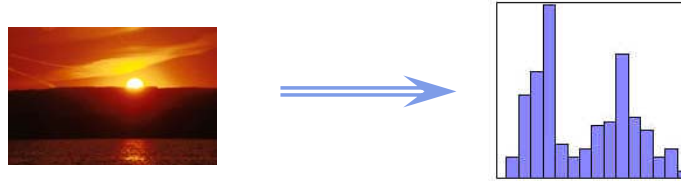


- Traditional approach to similarity search
- Deficiencies of the traditional approach
- Our proposal: the n-match query
- Algorithms to process the n-match query
- Experimental results
- Conclusions and future work



# Similarity Search : Traditional Approach

- **Objects represented by multidimensional vectors**



Elevation	Aspect	Slope	Hillshade (9am)	Hillshade (noon)	Hillshade (3pm)	...
2596	51	3	221	232	148	
...						

- **The traditional approach to similarity search: kNN query**

$$Q = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1)$$

ID	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	Dist
P1	1.1	1	1.2	1.6	1.1	1.6	1.2	1.2	1	1	<b>0.93</b>
P2	1.4	1.4	1.4	1.5	1.4	1	1.2	1.2	1	1	<b>0.98</b>
P3	1	1	1	1	1	1	2	1	2	2	<b>1.73</b>
P4	20	20	21	20	22	20	20	19	20	20	<b>57.7</b>
P5	19	21	20	20	20	21	18	20	22	20	<b>60.5</b>
P6	21	21	18	19	20	19	21	20	20	20	<b>59.8</b>



# Deficiencies of the Traditional Approach

- **Deficiencies**

- Distance is affected by a few dimensions with high dissimilarity
- Partial similarities can not be discovered

- **The traditional approach to similarity search: kNN query**

$$Q = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1)$$

ID	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	Dist
P1	1.1	100	1.2	1.6	1.1	1.6	1.2	1.2	1	1	99.0
P2	1.4	1.4	1.4	1.5	1.4	100	1.2	1.2	1	1	99.0
P3	1	1	1	1	1	1	2	100	2	2	99.0
P4	20	20	21	20	22	20	20	19	20	20	57.7
P5	19	21	20	20	20	21	18	20	22	20	60.5
P6	21	21	18	19	20	19	21	20	20	20	59.8

# The $N$ -Match Query : Warm-Up



- **Description**
  - **Matches** between two objects in  $n$  dimensions. ( $n \leq d$ )
  - The  $n$  dimensions are chosen **dynamically** to make the two objects **match best**.
- **How to define a “match”**
  - Exact match
  - Match with tolerance  $\delta$
- **The similarity search example**  
 $Q = ( 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 )$        $n = 6$

ID	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	Dist
P1	1.1	<b>100</b>	1.2	1.6	1.1	1.6	1.2	1.2	1	1	0.2
P2	1.4	1.4	1.4	1.5	1.4	<b>100</b>	1.2	1.2	1	1	0.4
P3	1	1	1	1	1	1	2	<b>100</b>	2	2	0
P4	20	20	21	20	22	20	20	19	20	20	19
P5	19	21	20	20	20	21	18	20	22	20	19
P6	21	21	18	19	20	19	21	20	20	20	19



# The $N$ -Match Query : The Definition

- **The  $n$ -match difference**

Given two  $d$ -dimensional points  $P(p_1, p_2, \dots, p_d)$  and  $Q(q_1, q_2, \dots, q_d)$ , let  $\delta_i = |p_i - q_i|, i=1, \dots, d$ . Sort the array  $\{\delta_1, \dots, \delta_d\}$  in increasing order and let the sorted array be  $\{\delta_1', \dots, \delta_d'\}$ . Then  $\delta_n'$  is the  **$n$ -match difference** between  $P$  and  $Q$ .

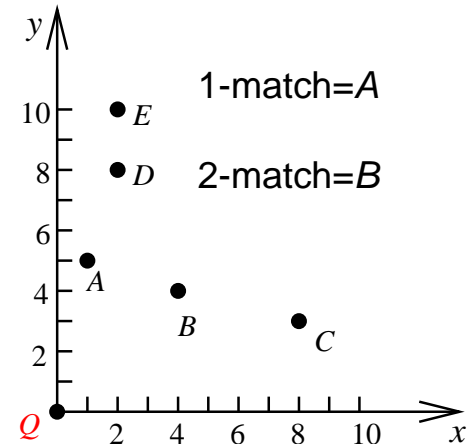
- **The  $n$ -match query**

Given a  $d$ -dimensional database  $DB$ , a query point  $Q$  and an integer  $n$  ( $n \leq d$ ), find the point  $P \in DB$  that has the smallest  $n$ -match difference to  $Q$ .  $P$  is called the  **$n$ -match** of  $Q$ .

- **The similarity search example**

$$Q = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1)$$

$$n = 6$$



ID	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	Dist
P1	1.1	<b>100</b>	1.2	1.6	1.1	1.6	1.2	1.2	1	1	0.6
P2	1.4	1.4	1.4	1.5	1.4	<b>100</b>	1.2	1.2	1	1	0.4
P3	1	1	1	1	1	1	2	<b>100</b>	2	2	1
P4	20	20	21	20	22	20	20	19	20	20	19
P5	19	21	20	20	20	21	18	20	22	20	19
P6	21	21	18	19	20	19	21	20	20	20	19



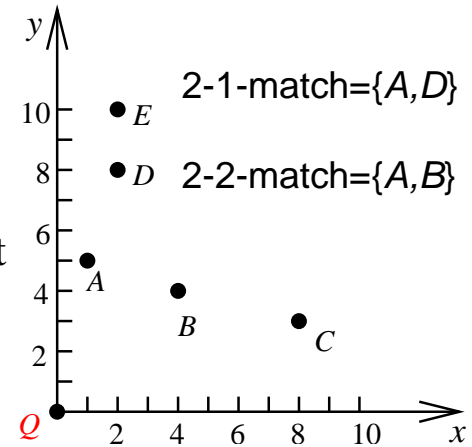
# The $N$ -Match Query : Extensions

- **The  $k$ - $n$ -match query**

Given a  $d$ -dimensional database  $DB$ , a query point  $Q$ , an integer  $k$ , and an integer  $n$ , find a set  $S$  which consists of  $k$  points from  $DB$  so that for any point  $P1 \in S$  and any point  $P2 \in DB-S$ ,  $P1$ 's  $n$ -match difference is smaller than  $P2$ 's  $n$ -match difference.  $S$  is called the  **$k$ - $n$ -match** of  $Q$ .

- **The frequent  $k$ - $n$ -match query**

Given a  $d$ -dimensional database  $DB$ , a query point  $Q$ , an integer  $k$ , and an integer range  $[n_0, n_1]$  within  $[1, d]$ , let  $S_0, \dots, S_i$  be the answer sets of  $k$ - $n_0$ -match,  $\dots$ ,  $k$ - $n_1$ -match, respectively, find a set  $T$  of  $k$  points, so that for any point  $P1 \in T$  and any point  $P2 \in DB-T$ ,  $P1$ 's number of appearances in  $S_0, \dots, S_i$  is larger than or equal to  $P2$ 's number of appearances in  $S_0, \dots, S_i$ .



- **The similarity search example**

$$Q = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1)$$

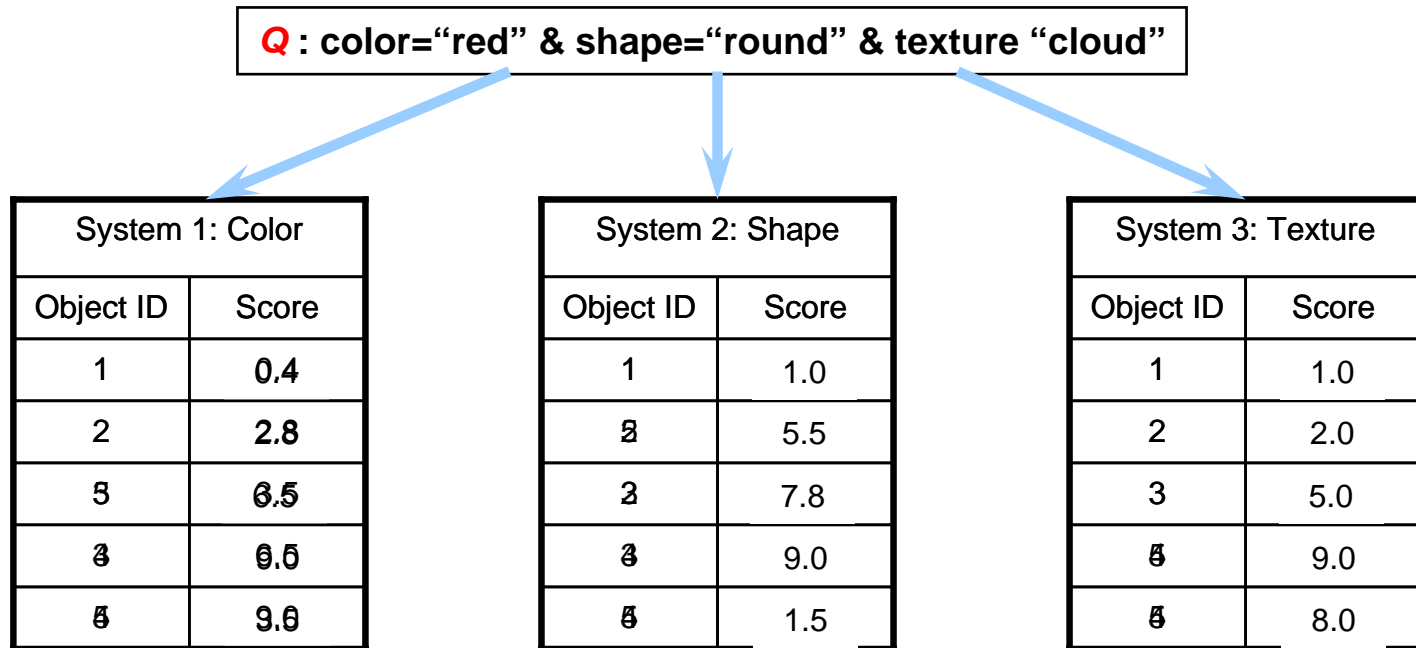
$$n = 6$$

ID	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	Dist
P1	1.1	<b>100</b>	1.2	1.6	1.1	1.6	1.2	1.2	1	1	0.2
P2	1.4	1.4	1.4	1.5	1.4	<b>100</b>	1.2	1.2	1	1	0.4
P3	1	1	1	1	1	1	2	<b>100</b>	2	2	0
P4	20	20	21	20	22	20	20	19	20	20	19
P5	19	21	20	20	20	21	18	20	22	20	19
P6	21	21	18	19	20	19	21	20	20	20	19



# Cost Model

- **The multiple system information retrieval model**
  - Objects are stored in different systems and scored by each system
  - Each system can sort the objects according to their scores
  - A query retrieves the scores of objects from different systems and then combine them using some aggregation function



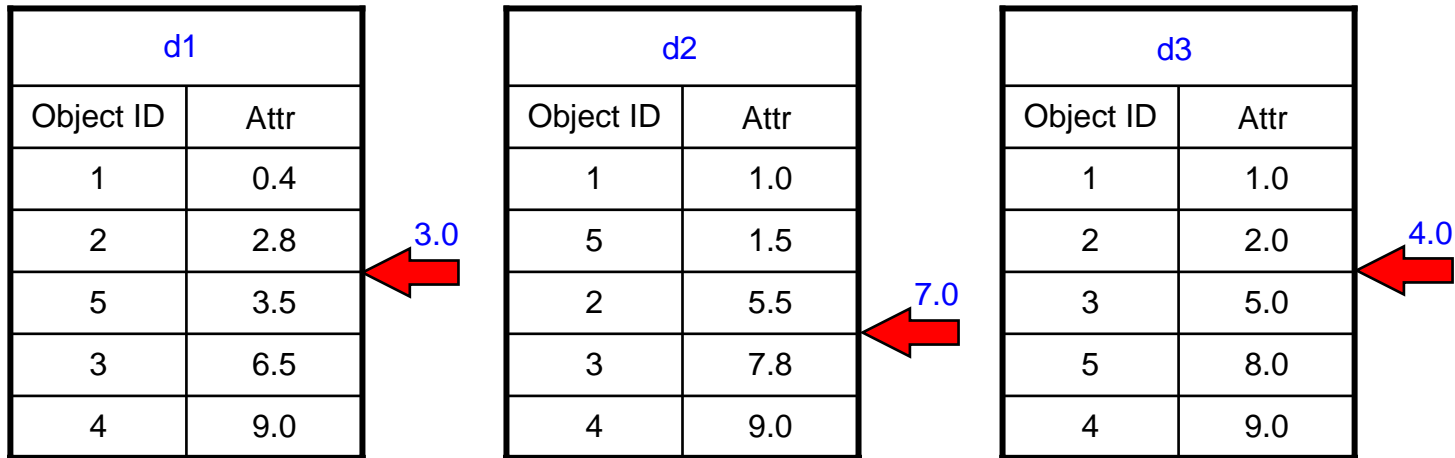
- **The cost**
  - Retrieval of scores – proportional to the number of scores retrieved
- **The goal**
  - To minimize the scores retrieved



# The AD Algorithm

- The AD algorithm for the  $k$ - $n$ -match query
  - Locate the query's attributes in every dimension
  - Retrieve the objects' attributes from the query's attributes in both directions
  - The objects' attributes are retrieved in **A**scending order of their **D**ifferences to the query's attributes. An  $n$ -match is found when it appears  $n$  times.

Q : color=2.2, distance=7.0, texture="cloud"  
 2 matches (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)



## Auxiliary structures

- Next attribute to retrieve  $g[2d]$

d1		d2		d3	
1, 2.6	3, 3.5	2, 1.5	4, 2.0	2, 2.0	5, 4.0

- Number of appearances  $appear[c]$

1	2	3	4	5
0	2	2	0	1

- Answer set  $S$        $\{3, 2\}$

# The AD Algorithm : Extensions



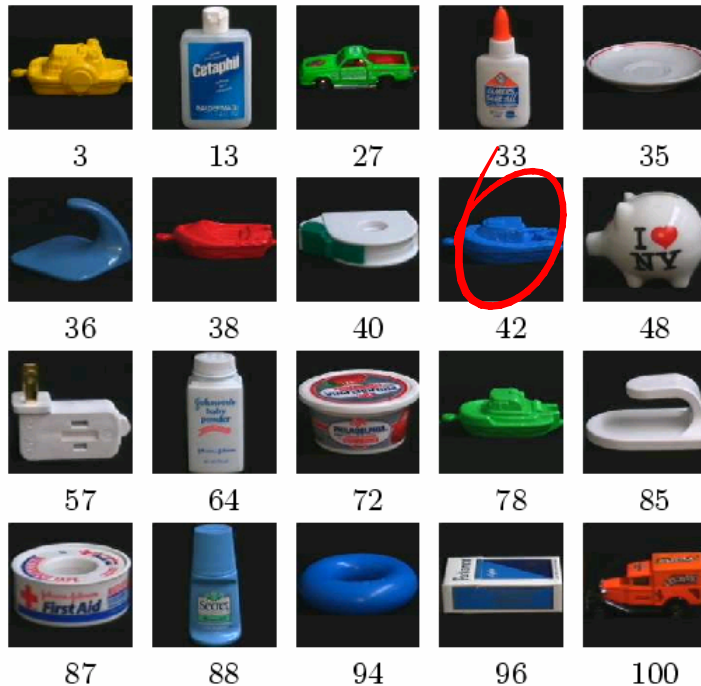
- **The AD algorithm for the frequent  $k$ - $n$ -match query**
  - The frequent  $k$ - $n$ -match query
    - Given an integer range  $[n_0, n_1]$ , find  $k$ - $n_0$ -match,  $k$ - $(n_0+1)$ -match, ... ,  $k$ - $n_1$ -match of the query,  $S_0, S_1, \dots, S_j$ .
    - Find  $k$  objects that appear most frequently in  $S_0, S_1, \dots, S_j$ .
  - Retrieve the same number of attributes as processing a  $k$ - $n_1$ -match query.
- **Disk based solutions for the (frequent)  $k$ - $n$ -match query**
  - Disk based AD algorithm
    - Sort each dimension and store them sequentially on the disk
    - When reaching the end of a disk page, read the next page from disk
  - Existing indexing techniques
    - Tree-like structures: R-trees, k-d-trees
    - Mapping based indexing: space-filling curves, iDistance
    - Sequential scan
    - Compression based approach (VA-file)



# Experiments : Effectiveness

- **Searching by  $k$ - $n$ -match**

- COIL-100 database
- 54 features extracted, such as color histograms, area moments



$k$ - $n$ -match query, $k=4$	
$n$	Images returned
5	36, 42, 78, 94
10	27, 35, 42, 78
15	3, 38, 42, 78
20	27, 38, 42, 78
25	35, 40, 42, 94
30	10, 35, 42, 94
35	35, 42, 94, 96
40	35, 42, 94, 96
45	35, 42, 94, 96
50	35, 42, 94, 96

$k$ NN query	
$k$	Images returned
10	13, 35, 36, 40, 42 64, 85, 88, 94, 96

- **Searching by frequent  $k$ - $n$ -match**

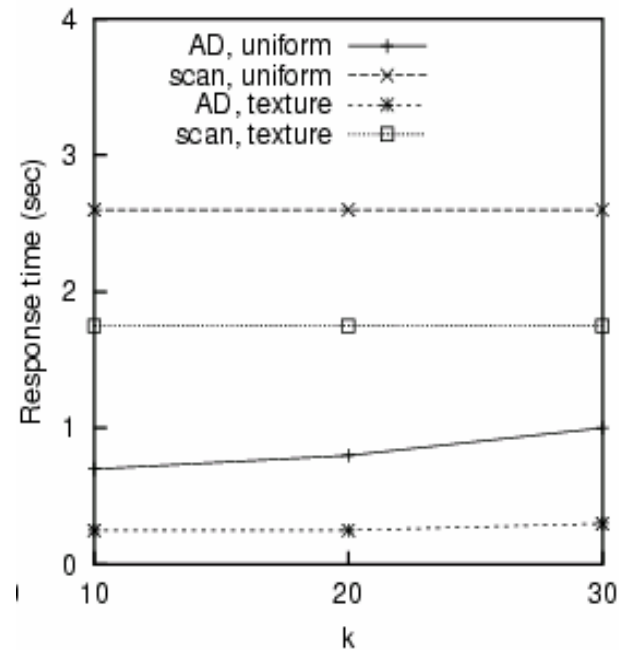
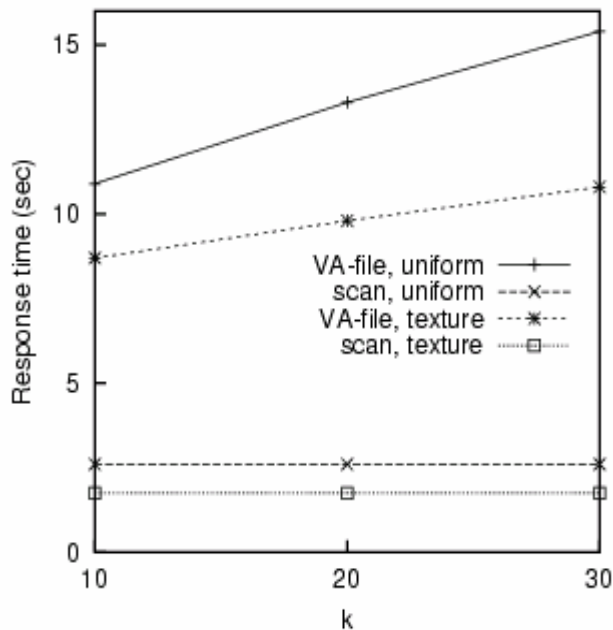
- UCI Machine learning repository
- Competitors:
  - IGrid
  - Human-Computer Interactive NN search (HCINN)

Data sets ( $d$ )	IGrid	HCINN	Freq. $k$ - $n$ -match
Ionosphere (34)	80.1%	86%	87.5%
Segmentation (19)	79.9%	83%	87.3%
Wdbc (30)	87.1%	N.A.	92.5%
Glass (9)	58.6%	N.A.	67.8%
Iris (4)	88.9%	N.A.	89.6%



# Experiments : Efficiency

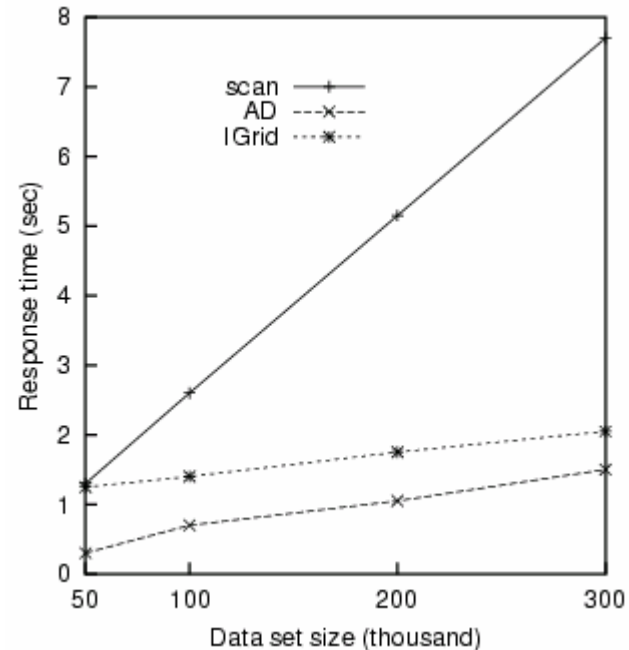
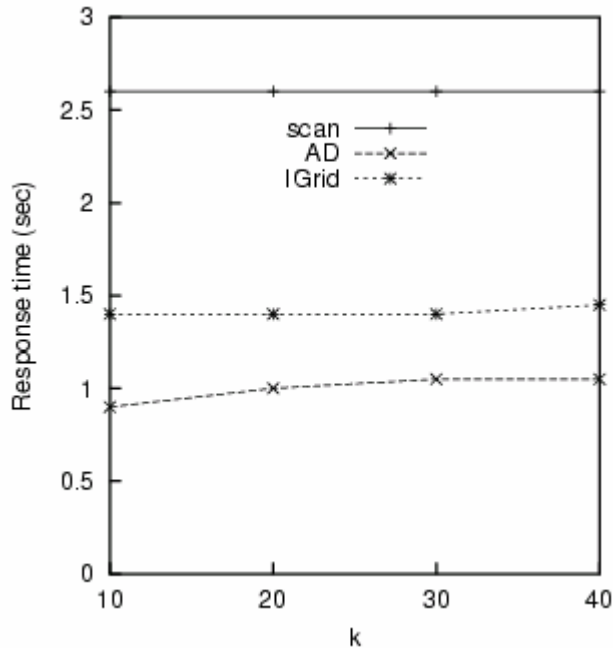
- **Disk based algorithms for the Frequent  $k$ - $n$ -mach query**
  - Texture dataset (68,040 records); uniform dataset (100,000 records)
  - Competitors:
    - The AD algorithm
    - VA-file
    - Sequential scan





# Experiments : Efficiency (continued)

- **Comparison with other similarity search techniques**
  - Texture dataset ; synthetic dataset
  - Competitors:
    - Frequent  $k$ - $n$ -match query using the AD algorithm
    - IGrid
    - Human-Computer Interactive NN search ([HCINN](#))



# Conclusions and Future Work



- **Conclusions**

- We proposed a new approach to do similarity search, that is, the  $k$ - $n$ -match query. It has the advantage of being tolerant to noise and able to discover partial similarity.
- If we don't choose a good  $n$  value, the results of the  $k$ - $n$ -match query may not be good enough to find full similarity, so we further propose the frequent  $k$ - $n$ -match query to address this problem.
- We proposed the AD algorithm, which is optimal for both the  $k$ - $n$ -match query and the frequent  $k$ - $n$ -match query under the multiple system information retrieval model. We also apply it in a disk based model.
- Based on an extensive experimental study, we see that the frequent  $k$ - $n$ -match query is more effective in similarity search than existing techniques such as IGrid and Human-Computer Interactive NN search. We also see that the frequent  $k$ - $n$ -match query can be processed more efficiently than other techniques by our proposed AD algorithm in a disk based model.

- **Future work**

- We may perform more experiments to see whether the traditional  $k$ NN search can always be replaced by frequent  $k$ - $n$ -match search; if not, in which scenarios we should use it?



# Questions?

## My contact

- Email: [rui@csse.unimelb.edu.au](mailto:rui@csse.unimelb.edu.au)
- Website: <http://www.csse.unimelb.edu.au/~rui>