

# QUERY PROCESSING AND OPTIMIZATION FOR PICTORIAL QUERY TREES

HANAN SAMET

COMPUTER SCIENCE DEPARTMENT AND  
CENTER FOR AUTOMATION RESEARCH AND  
INSTITUTE FOR ADVANCED COMPUTER STUDIES  
UNIVERSITY OF MARYLAND

COLLEGE PARK, MARYLAND 20742-3411 USA

Joint work with Aya Soffer.

These notes may not be reproduced by any means (mechanical or electronic or any other) without express written permission of Hanan Samet

## INTRODUCTION

- Problem: find all images in a database of images that contain particular objects in a specific spatial configuration with respect to each other
- Solution: SQL extension with predicates corresponding to spatial relationships
- Drawback: must pre-classify all objects in image so user can specify them by some alphanumeric tag
- Alternative: pictorial specification
  1. use implicit characteristics of the pictorial query
    - more natural
  2. difficulties:
    - how do we know if an object in the database image is really the same as the object in the query image (matching ambiguity)?
    - if query image has several objects, will a subset be good enough or do we want to find all of them (contextual ambiguity)?
    - is the spatial arrangement of the objects in the query image important (spatial ambiguity)?
  3. not always as expressive as textual queries in the specification of combinations of conditions and negative conditions such as finding all images containing beaches but no camping sites within three miles of these beaches





## EXISTING WORK

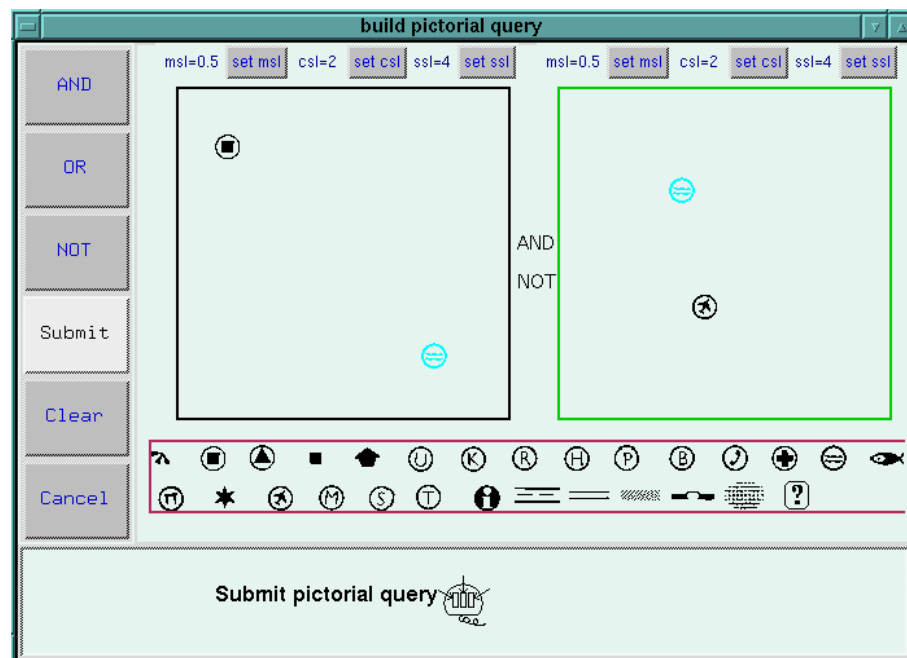
1. Usually global image matching based on color and texture features (e.g., QBIC)
2. Ambiguity of matching one query image object to another (e.g., Grosky and Mehrotra, Del Bimbo)
3. Don't usually deal with queries involving several objects and their spatial configuration
4. Only address matching ambiguity and do not deal with spatial and contextual ambiguity
5. Some work on specifying topological and directional relations among query objects
  - focus on defining and efficiently computing spatial relations between objects stored in a database
  - only deal with tagged images - do not address matching ambiguity
  - try to match as many query objects as possible
  - usually assume relative locations of objects are exactly as in the query image (e.g., Gudivada)
6. Limited form of spatial ambiguity permitted in 2D-string and variants
  - do not address distance between objects
  - assume database image must contain all objects in query image
  - do not deal with matching or contextual ambiguity
7. Do not permit Boolean combinations or negation of query images


## MECHANICS OF PICTORIAL QUERY SPECIFICATION

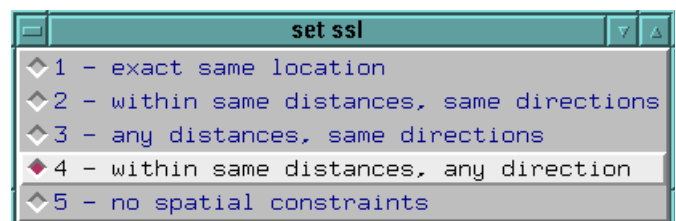
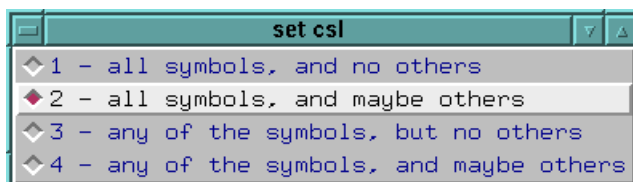
- Pictorial query composed of one or more query images
- Query image is built by selecting symbols that should appear in database of images from a menu of symbols and position them by dragging so that the desired spatial constraints hold
- Specify image similarity level to satisfy matching, contextual, and spatial constraints between query image and database images
- Can compose more complex queries by use of AND, OR, and NOT
- Can use object binding to specify that two query-image symbols must match the same instance of a database image
  1. could also permit several instances
  2. only useful when AND as irrelevant when OR
  3. object binding is achieved by dragging subsequent instances from the first query image rather than dragging them from the menu of symbols

## EXAMPLE





- Retrieve all database images that have a hotel  within 6 miles of beach  and do not have an airport  within 1 mile of the beach  (with certainty of 0.5 that the database image symbols are in fact a hotel, beach, and airport)



- Use color coding to denote that two query-image beach  symbols are bound to the same instance in the database image
- Contextual similarity menu: Spatial similarity menu:



## SYMBOL DICTIONARY

 airport	 fishing site	 camping site
 gas station	 beach	 hotel
 cafe'	 first aid	 site of interest
 restaurant	 museum	 wild card
 picnic site	 open field	
 post office	 railroad	 two-lane road
 scenic view	 local road	 one-lane road

## SYSTEM OVERVIEW

- Query interface for MARCO (MAp Retrieval by COntent)
- Input to MARCO
  1. raster images of separate map layers
  2. raster images of map composites (result of composing separate map layers)
  3. used tourist symbol layer which contained geographic symbols representing campsites, beaches, hotels, etc.
  4. recognition driven by information in legend
- Input process requires user intervention to build an initial training set
- Used training set library to assign candidate classifications to each symbol using weighted bounded several-nearest neighbor classifier
- Assigned certainty value between 0 and 1 to each classification to indicate its certainty
- There can be more than one possible classification in which case the database stores the different classifications with their certainty
- Classification is based on set of features describing symbol's shape thereby enabling resolution of matching ambiguity

## OUR APPROACH

- User specifies required level of similarity as part of the pictorial query
- Query image is constructed by:
  1. creating a query image containing the required objects positioned so that desired spatial constraints hold
  2. specifying desired level of similarity in three domains:
    - a. *matching similarity* : certainty of classifications of matching symbols
    - b. *contextual similarity* : how well does the content of one image match that of another
    - c. *spatial similarity* : the relative locations and orientations of the matching symbols in the two images
- All images similar to the query image under the specified similarity levels are retrieved
- Definitions:
  1. *symbol* : a group of connected pixels that have some semantic meaning
  2. *class* : a group of symbols that all have the same semantic meaning
  3. symbol  $s_1$  matches symbol  $s_2$  if they belong to the same class

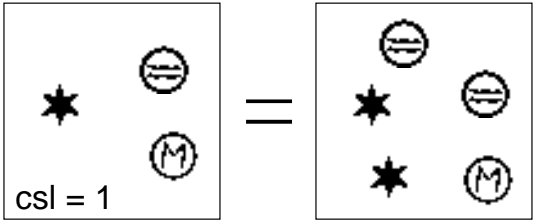
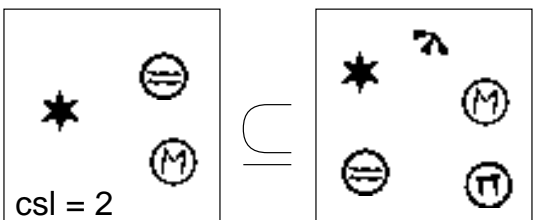
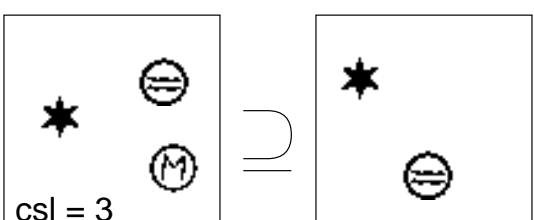
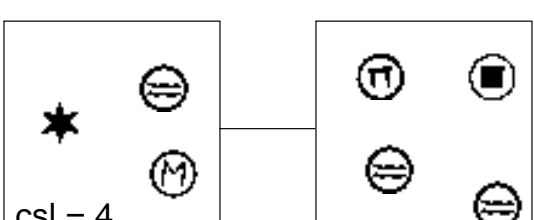


## MATCHING SIMILARITY LEVEL (MSL)

- Specifies how close a match between a symbol  $s_1$  in query image  $Q/$  and a symbol  $s_2$  in database image  $D/$  is needed for them to be considered the same
- A number between 0 and 1 indicating a lower bound on the certainty that two symbols are from the same class

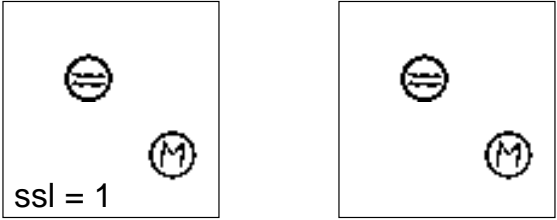
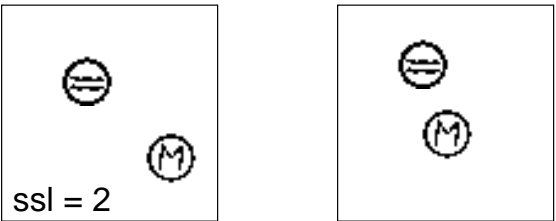
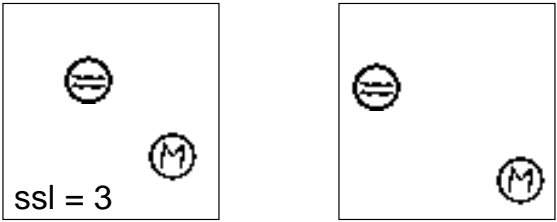
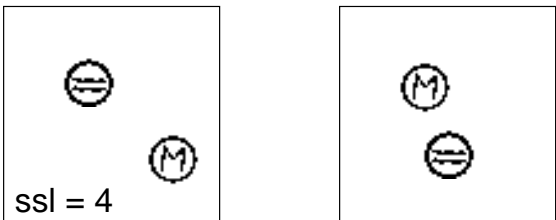
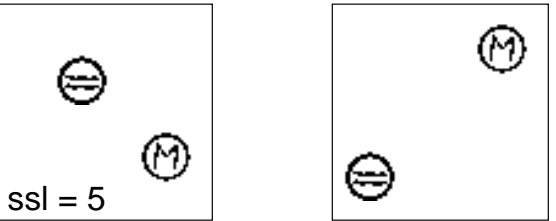
## CONTEXTUAL IMAGE SIMILARITY (CSL)

- Four levels of contextual similarity between  $QI$  and  $DI$ :

<p>1. Every symbol in <math>QI</math> has a matching symbol in <math>DI</math>, and every symbol in <math>DI</math> has a matching symbol in <math>QI</math></p>	 <p>QI DI</p>
<p>2. Every symbol in <math>QI</math> has a distinct matching symbol in <math>DI</math> (<math>DI</math> may contain additional symbols from any class)</p>	 <p>QI DI</p>
<p>3. Every symbol in <math>DI</math> has a matching symbol in <math>QI</math></p>	 <p>QI DI</p>
<p>4. At least one symbol in <math>QI</math> has a matching symbol in <math>DI</math> (<math>DI</math> may contain additional symbols from any class)</p>	 <p>QI DI</p>

## SPATIAL IMAGE SIMILARITY (SSL)

- Five levels of spatial similarity between  $DI$  and  $QI$ :

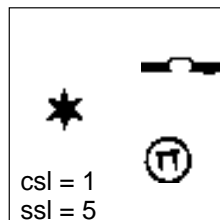
1. Matching symbols are in exact same locations	
2. Same spatial relationship between matches and distance between symbols in $DI > 0$ and $\leq$ distance between matches in $QI$  icprf5: Images with a hotel within 10 of a beach AND a scenic view northeast and within 9 of the beach	 <p style="text-align: center;">QI                      DI</p>
3. Same spatial relationship between matches but distance between matches may vary	 <p style="text-align: center;">OI                      DI</p>
4. Spatial relationship between matches varies; distance between symbols in $DI > 0$ and $\leq$ distance between matches in $QI$  icpr5_d: Changle SSL in icprf5 to 4 (No Spatial relationship)	 <p style="text-align: center;">QI                      DI</p>
5. Distance and spatial relationship between matching symbols may vary (no spatial constraints)	

## TOTAL IMAGE SIMILARITY

1. Total similarity between  $Q/$  and  $D/$ : combine the three similarity factors
2. Ex:  $D/ \equiv_{0.5,2,3} Q/$  denotes matching similarity (msl) at level 0.5, contextual similarity (csl) at level 2, and spatial similarity (ssl) at level 3
  - for each symbol in  $Q/$  there is a matching symbol with certainty 0.5 from the same class in  $D/$
  - location of the symbols and the distance between them may vary
  - directional inter-symbol spatial relationship between them is the same
3. General case:
  - $D/ \equiv_{msl,csl,ssl} Q/$  and
  - $S' = \{\text{symbols of } D/ \text{ that match a symbol in } Q/\}$
  - implies:
    - a. set of classes of symbols of  $S'$  is a subset of the set of classes of the symbols of  $Q/$
    - b. for every pair of symbols  $s_1, s_2 \in S'$  the spatial constraints dictated by ssl hold

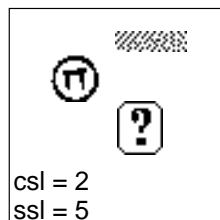
## EXAMPLE QUERIES VARYING CSL

- Assume ssl=5 (no spatial constraints)



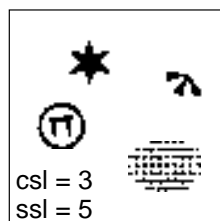
Q1

images that contain a site of interest ★, a picnic site (π), a railroad —, and no symbols from other classes



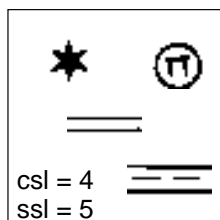
Q2

images that contain a picnic site (π), a local road —, and at least one other arbitrary symbol (wild card); there may symbols from other classes



Q3

images that contain a site of interest ★, or a scenic view ↷, or a picnic site (π), or an open field — (an image may contain any combination but not symbols from other classes)

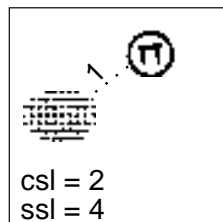


Q4

images that contain a site of interest ★, or a picnic site (π), or a one-lane road —, or a two-lane road — (an image may contain one or all of them as well as other symbols)

## EXAMPLE QUERIES VARYING SSL

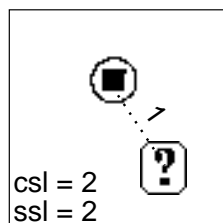
- Assume  $csl=2$  so every symbol in query image ( $Q$ ) has matching symbol in the database image
- Distances derived implicitly from actual distance in  $Q$



Q1

images that contain a picnic site  $\textcircled{P}$  within 1 mile of an open field  $\textcircled{F}$

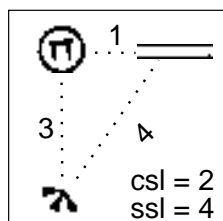
**icprf4:** Images with a camping site within 5 of a beach or hotel within 10 of a beach.



Q2

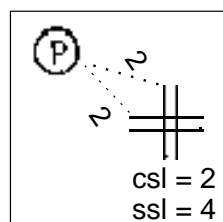
images that contain a hotel  $\textcircled{H}$  and a symbol of any type  $\textcircled{?}$  (wild card) within 1 mile and southeast of this hotel  $\textcircled{H}$  (there may be more symbols)

**beach\_+\_anything:** Images with a beach and anything (wildcard) within 20



Q3

images that contain a picnic site  $\textcircled{P}$  within 1 mile of a one-lane road  $\text{==}$  and within 3 miles of a scenic view  $\textcircled{V}$  and requires that the scenic view  $\textcircled{V}$  be within 4 miles of the one-lane road  $\text{==}$

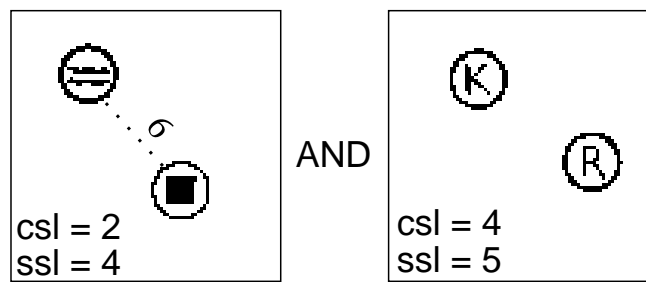


Q4

images that contain a post office  $\textcircled{P}$  within 2 miles of two one-lane roads  $\text{==}$  that intersect (distance between  $\text{==}$  is zero)

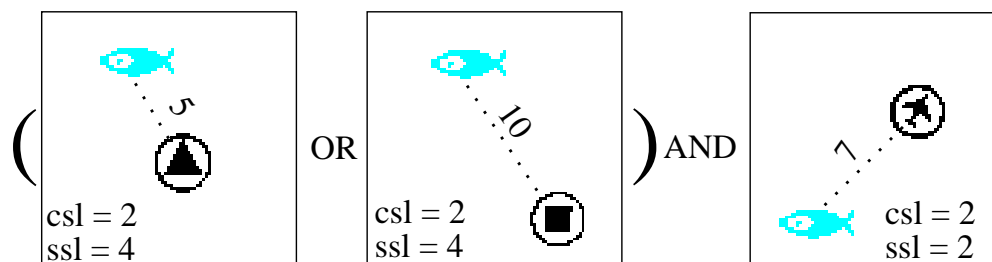
## EXAMPLE COMPOUND QUERIES







- Compound queries are specified by combining query images with “AND” and “OR” operators
- Example 1:



all images with a hotel  within 6 miles of a beach  AND with a cafe  or a restaurant 

- Example 2:

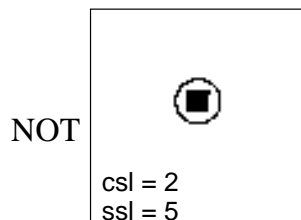


all images with a camping site  within 5 miles of a fishing site  OR with a hotel  within 10 miles of a fishing site  AND with an airfield  northeast of and within 7 miles of the fishing site 

## QUERIES WITH NEGATION

- Specify conditions that should not be satisfied by database images that are retrieved successfully
- Useful to constrain images that do not contain a particular symbol, a pair of symbols, or one of two symbols

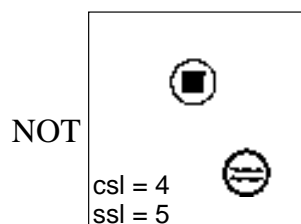
1.





Q1

all images with no hotel 

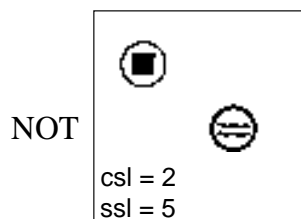
2.





Q2

all images that do not have a beach  and do not have a hotel 

3.



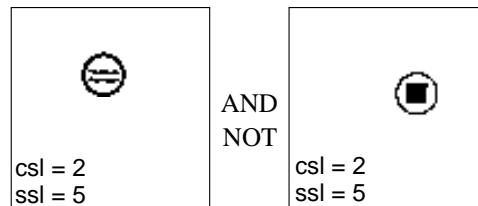
Q3

all images that do not have a beach  or do not have a hotel 



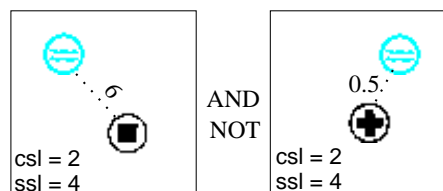
## NEGATION WITH COMPOUND QUERIES





1. Can specify both positive and negative conditions



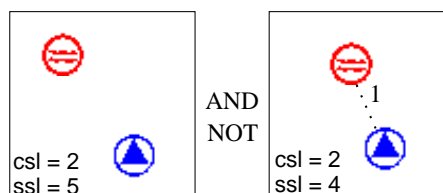
all images that have a beach  but not a hotel 



2. Can specify more than one spatial condition for the same symbol by use of object binding and negation



images with a hotel  within 6 miles of a beach  and with no first aid station  within 0.5 miles of the beach 

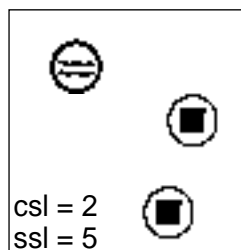
3. Can specify distance constraints in terms of an upper bound using symbol binding and negation








all images with a camping site  further than one mile from the beach 

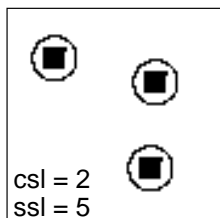
## MULTIPLE INSTANCES OF A SYMBOL IN THE QUERY IMAGE


- Useful to specify number of occurrences of a particular symbol in the target database image
- If  $csl$  is 1 or 2, then each symbol in  $Q/I$  must have a distinct matching symbol in  $D/I$  for  $D/I$  to satisfy the query

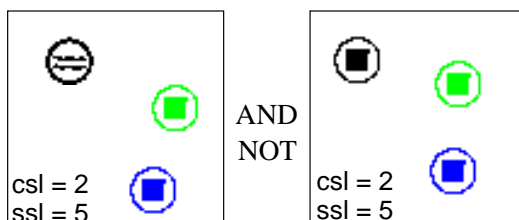




all images with a beach  
 and at least two hotels  


- If want to restrict ourselves to a beach and exactly two hotels, then create a compound query of finding all images with a beach  and at least two hotels  with the negation of the query that finds all images with at least three hotels 



all images with at least  
three hotels 



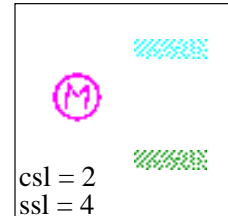
all images with a beach   
and exactly two hotels 

<beach+atleast\_2hotel>

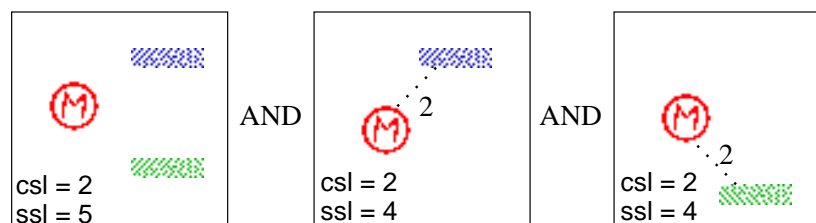
## MULTIPLE INSTANCES OF A SYMBOL IN THE QUERY IMAGE WITH DIFFERENT SPATIAL CONSTRAINTS

1. Spatial constraints must hold for all of the symbols in the query image or for none of them
2. Ex: all images with two different local roads within 2 miles of a museum

- use  $csl=2$  as this permits the database image to contain symbols from other classes



- above query does not work as it requires that the two local roads are also within 2 miles of each other
- need to have a conjunction (i.e., AND) of separate query images with object binding to ensure that they use the same instances of the symbols in the database image









left: specify contextual similarity condition

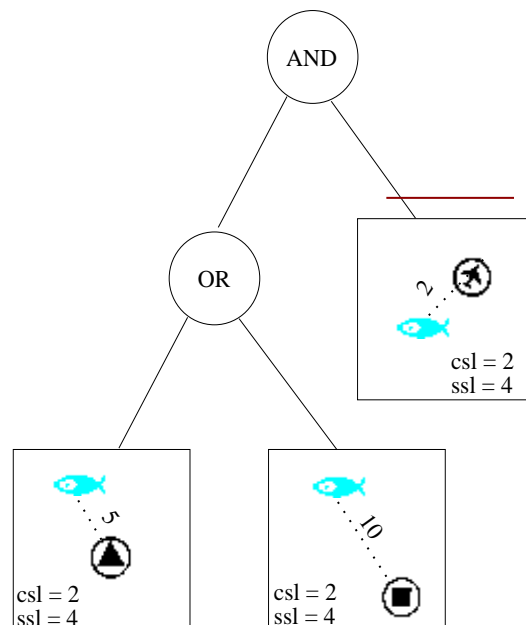
center: specify spatial similarity constraint

between museum (M) and one local road

right: specify spatial similarity constraint between museum (M) and second local road







## PICTORIAL QUERY TREES

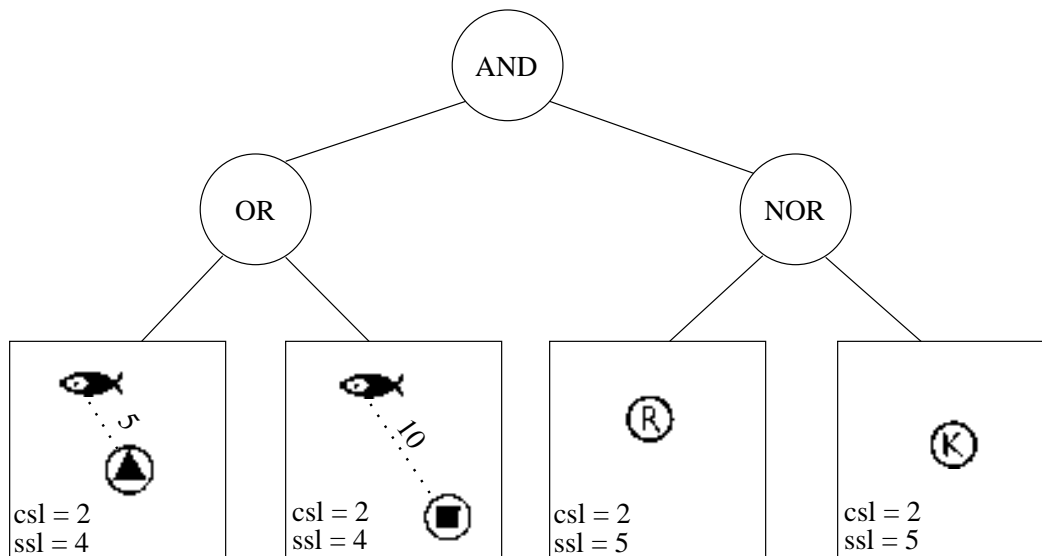
1. Leaves are individual pictorial queries
2. Internal nodes represent AND, OR, XOR, and their negations
3. Can use object binding to stipulate the same instances of a symbol in different leaf nodes
4. Ex: Images with a camping site  within 5 miles of a fishing site  OR with a hotel  within 10 miles of a fishing site  AND with no airport  within 2 miles of the fishing site 



- notice the use of negation via a bar on the right son of the root
- notice the use of object binding
- csl = 2 so every symbol in the query image has a matching symbol in the database image

## EXAMPLE COMPLEX QUERY TREE

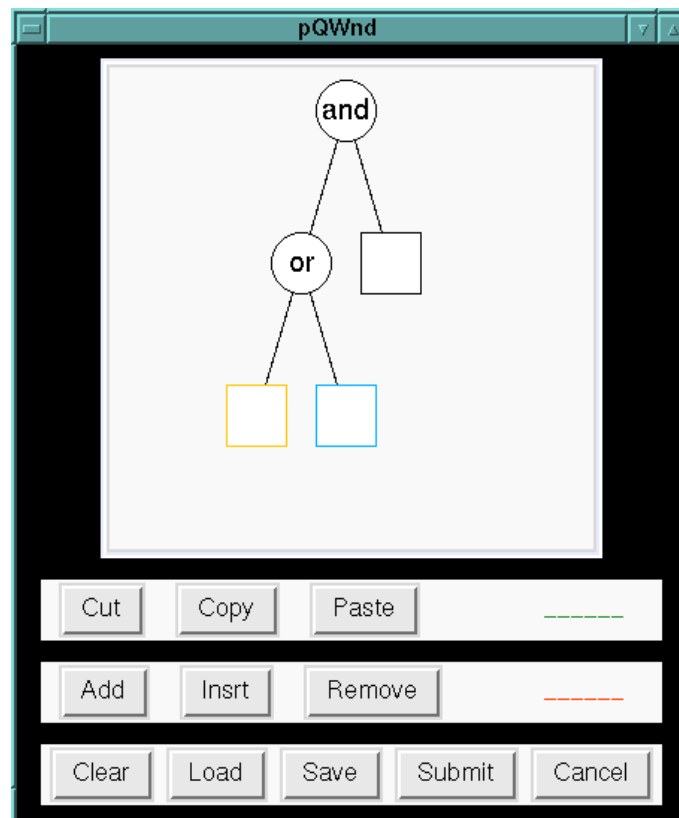
- Images with a camping site  within 5 miles of a fishing  site OR with a hotel  within 10 miles of a fishing  site AND with neither a restaurant  nor a cafe 



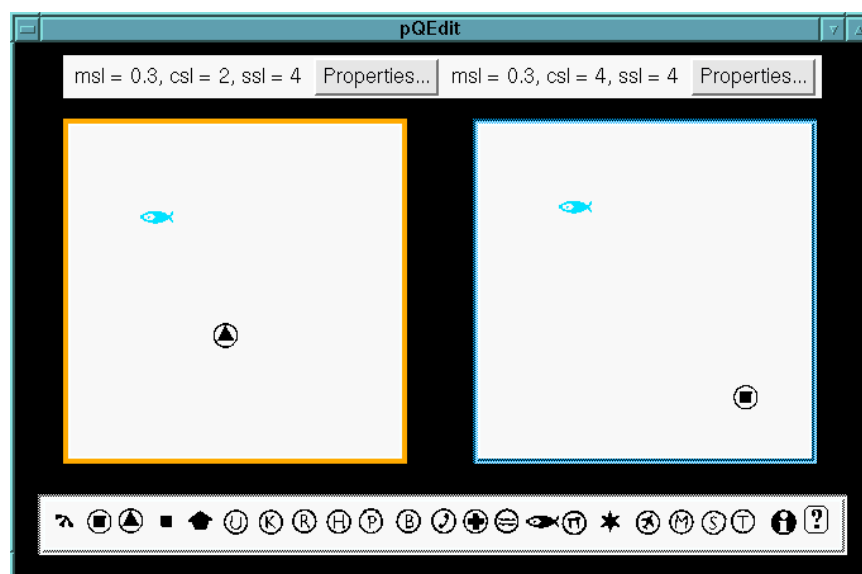
- no need for object binding
- internal node operator is negated (NOR)
- csl = 2 so every symbol in the query image has a matching symbol in the database image

## USER INTERFACE FOR PICTORIAL QUERY TREES

1. One window is used for building structure of the tree



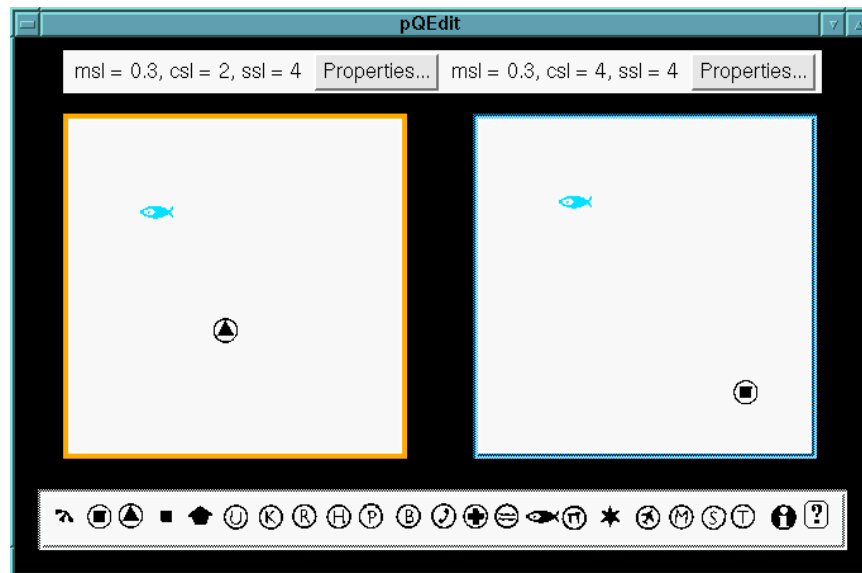
2. Second window for editing a pair of individual queries



## BUILDING THE QUERY TREE STRUCTURE

1. First row (Cut, Copy, Paste) is used to edit the query tree
  - pictorial query or query subtree can be “cut” or “copied” into a clipboard and then “pasted” into another part of the tree or into a different query tree
  - "Cut" removes query subtree from the query tree while "Copy" retains it in the original query tree
2. Second row (Add, Insert, Remove) adds leaf nodes (pictorial queries), inserts internal nodes (logical operators), and removes both and leaf and internal nodes
  - removal of internal nodes is only allowed when the node has exactly one child (this child replaces the internal node)
  - don't allow deletion of last child of any internal node
  - logical operators are selected by repeatedly clicking on node
  - middle mouse button negates selected internal node
3. Third row (Clear, Load, Save, Submit, Cancel) controls entire query tree
  - “Clear” the window by deleting everything
  - “Load” previously saved query tree
  - “Save” query tree in a file
  - “Submit” query tree for evaluation
  - “Cancel” exits pictorial query specification interface

## QUERY WINDOW FOR EDITING INDIVIDUAL QUERIES



1. Displays two leaf nodes from query tree
2. Color of border of query indicates correspondence with query tree
3. Move pictorial query from query tree to query window by clicking on it
4. Symbols are “dragged” and “dropped” from menu of symbols at bottom of query window
  - example symbols are from the legend of map
  - could also import examples of symbols directly
5. Object binding is achieved by dragging symbol from one of the two nodes to the other and they have the same non-black color
  - black symbols are not bound
6. Matching, contextual, and spatial similarity are set via the “Properties Menu”



## PICTORIAL QUERY PROCESSING

- Retrieve all database images conforming to pictorial query
- Assume database contains indexes to
  1. retrieve all images containing symbols from a given class
    - B-tree
  2. retrieve all symbols in a given image
    - B-tree
  3. retrieve all symbols within a given distance or direction from a given point
    - index on locations of the set of symbols from all images
    - use PMR quadtree for points

## PROCESSING A SINGLE PICTORIAL QUERY IMAGE (QI)

1. For each symbol, find all database images DI that contain it with appropriate msl value
2. If  $csl=1$  or  $2$  (i.e., we want images containing all symbols in QI), then take intersection of DI
3. If  $csl=3$  or  $4$  (i.e., one symbol from QI is enough), then take union of DI
4. If  $csl=1$  or  $3$  then eliminate any images from DI that contain symbols not in QI
5. If  $csl=1$  or  $2$ , check that multiple instances of a symbol in QI have at least that many instances in DI
6. check if spatial constraints match for each image in DI
  - need to check many possible matchings as allow multiple instances of a symbol in QI
  - for each symbol in QI find all possible matches in image I of DI (as multiple instances of a symbol are allowed in QI) making sure that no more than one symbol from QI is matched to any symbol in image I
  - select one element from each of the sets of matches for each symbol in QI and check actual spatial similarity using CheckSsl
7. Order all results by the average matching certainty of all matching symbols and return as result of query

## PROCESSING AND EVALUATING PICTORIAL QUERY TREE

1. If current node N is a leaf node, then apply single query image algorithm
  - if node is negated, then result is set of all images that do NOT satisfy the query
2. If current node N is a nonleaf node, then apply algorithm recursively to the children of N and then apply the logical operation
3. Results of processing leaf nodes are cached so no need to recompute if no change to a particular query
  - no caching of results for nonleaf node - cost of recomputing them is negligible as does not involve I/O
4. Object binding is handled by modifying the AND logical operation to make use of a global set of object bindings that were used in the construction of the query
  - OR and XOR do not make use of object bindings

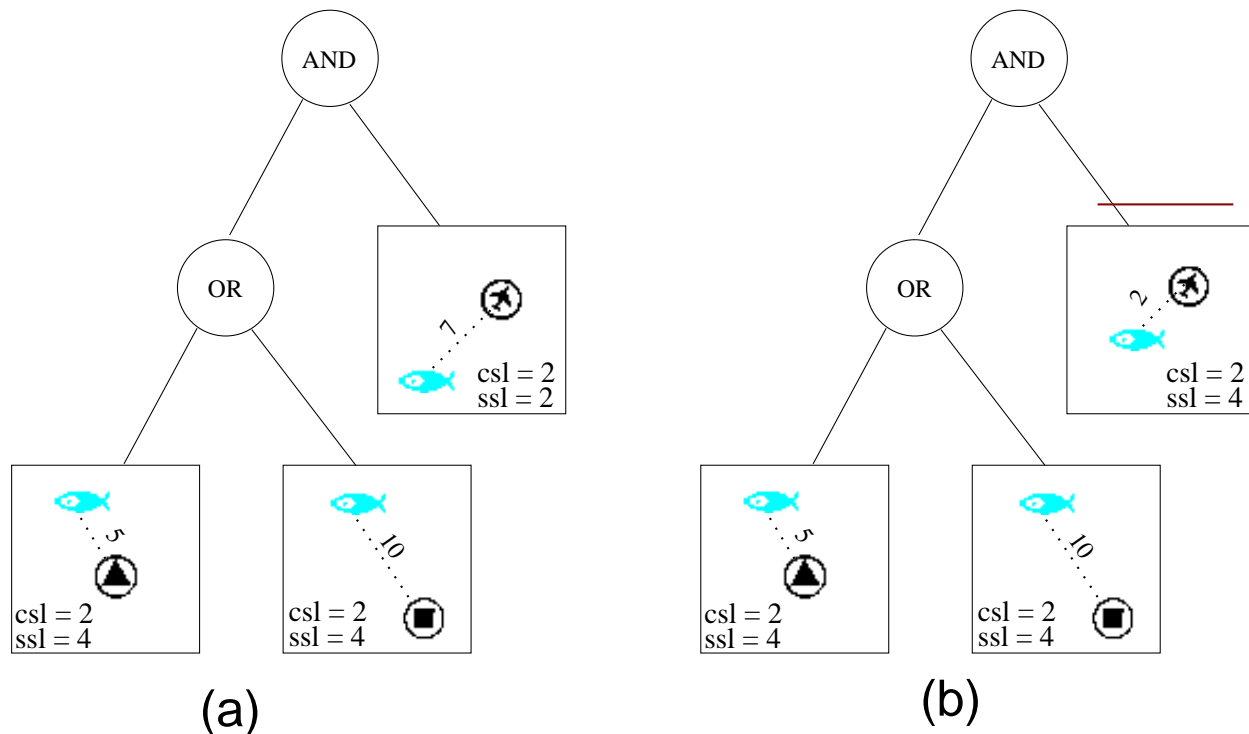
## QUERY OPTIMIZATION ISSUES











- The algorithm we presented for handling individual pictorial queries is naive
  1. symbol matching
  2. contextual constraints
  3. spatial constraints
- Can make some improvements
  1. handle contextual and spatial constraints simultaneously to achieve better pruning of search space early on
  2. make use of selectivity estimates to change order of processing of query images in pictorial query tree
  3. combine individual query images in pictorial tree and process together

## PICTORIAL QUERY TREE OPTIMIZATION VIA SELECTIVITY ESTIMATION

- Change order of processing of individual query images so that the more selective ones are processed first
  1. matching selectivity
    - estimates how many images satisfy the msl value
  2. contextual selectivity
    - estimates how many images satisfy the csl value
  3. spatial selectivity
    - estimates how many images satisfy the ssl value
- Matching and contextual selectivity factors depend on statistics
  1. stored as histograms in the database and indicate distribution of classifications and certainty levels in the images
  2. histograms constructed when populating database
- Computing spatial selectivity factors is complex
  1. measure distance aspect by calculating some approximation of area spanned by the symbols in query image
  2. e.g., convex hull of symbols in query image
- Query tree selectivity is computed by a recursive algorithm similar to the query processing algorithm
  1. negation implies subtracting query selectivity from 1
  2. OR and XOR: sum of selectivities of subtrees
  3. AND: product of selectivities of subtrees



## EXAMPLE OF USE OF SELECTIVITY ESTIMATION

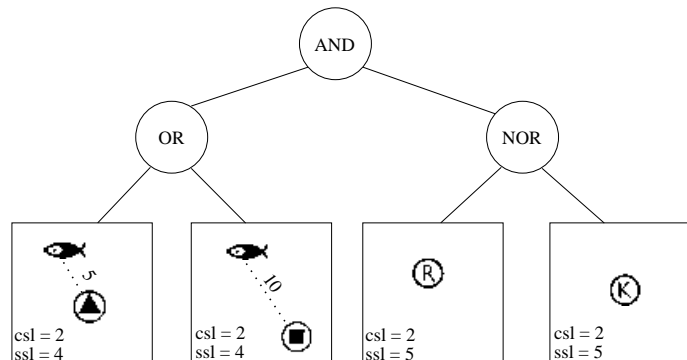


- Left side of tree seeks images with a camping site  within 5 miles of the fishing site  OR a hotel  within 10 miles of a fishing site 
- Right side of (a) has constraint that there is an airport  northeast of and within 7 miles of fishing site 
  - very few airports  in database implying the right side is more selective and is processed first
- Right side of (b) has constraint that there is no airport  within 2 miles of the fishing site 
  - very few airports  in database implying the right side is not selective and left side is processed first

# PICTORIAL QUERY TREE OPTIMIZATION VIA QUERY IMAGE COMBINATION

- Current algorithm:

1. CF: images with a camping site  within 5 miles of a fishing site 



2. HF: images with hotel  within 10 of fishing site 

3. LS = UNION of CF and HF


4. R: images with a restaurant 



5. C: images with a cafe 

6. RS = all images with exception of R and C


7. final result is intersection of LS and RS

- Better algorithm: form candidate list CL

1. for each fishing site  , find nearest neighbors up to 5 miles in incremental order








- if next nearest symbol is a camping site  or a hotel  , then add image to CL

2. continue retrieving nearest neighbors up to 10 miles

- if next nearest symbol is hotel  , add image to CL

3. remove images in CL with restaurant  or cafe 

## CONCLUDING REMARKS AND FUTURE WORK

- Need much more work on developing contextual and spatial selectivity estimation techniques
- Need to expand expressive power of queries
  1. can find post office  within 2 miles of two one-lane roads  that intersect but NOT that the post office  is within two miles of the point where the two one-lane roads  intersect
  2. although we take the extent of objects into account in distance and direction computations, we do not consider the size or direction of the object itself
    - we cannot specify an open field  whose area is at least one square mile
    - we cannot specify a local road  that goes from north to south
  3. include non-spatial conditions
    - cannot find hotels  whose price is less than \$80 per night
- Try out in other applications (e.g., medical)