

Probabilistic Algorithms for Matching Images¹

Helmut Alt Ludmila Scharf Sven Scholz

Institute of Computer Science
Freie Universität Berlin

April 23-28, 2006
Dagstuhl

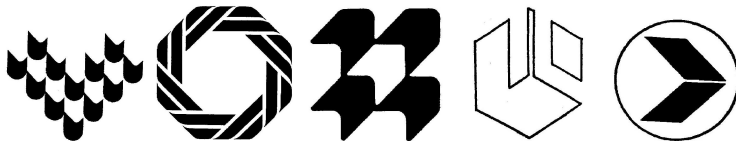
¹Supported by the European Union under contract No. FP6-511572, Project PROFI.

Project PROFI- Perceptually Relevant Retrieval of Figurative Images

Recognizing Similarities between **trademarks**

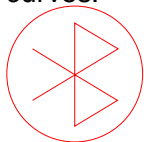
Project PROFI- Perceptually Relevant Retrieval of Figurative Images

Recognizing Similarities between **trademarks**

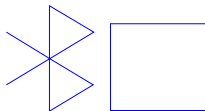


Matching Problem

Given: two shapes $A, B \subset \mathbb{R}^2$, modeled by a set of polygonal curves.



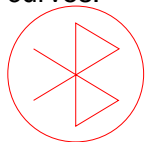
A



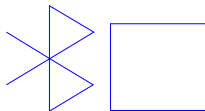
B

Matching Problem

Given: two shapes $A, B \subset \mathbb{R}^2$, modeled by a set of polygonal curves.

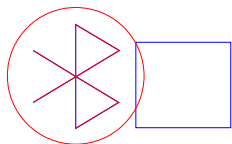


A



B

Find: a transformation t , such that $t(B)$ matches best A.

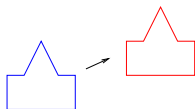


A

t(B)

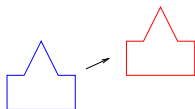
Transformations

- Translations

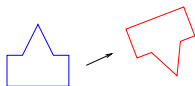


Transformations

- Translations

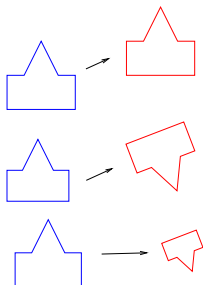


- Rigid Motions



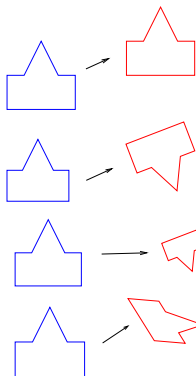
Transformations

- Translations
- Rigid Motions
- Similarities



Transformations

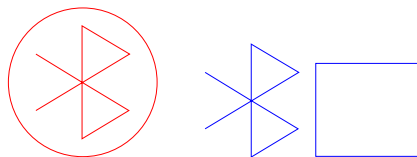
- Translations
- Rigid Motions
- Similarities
- Affine Maps



Probabilistic Matching

For matching A to B by translation:

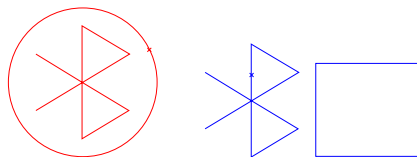
- Take random points $a \in A$ and $b \in B$ and insert vector $v = b - a$ into a sample of **translation space**
- Repeat this experiment to obtain an approximation of the probability distribution in translation space
- Possible candidates = centers of **clusters**



Probabilistic Matching

For matching A to B by translation:

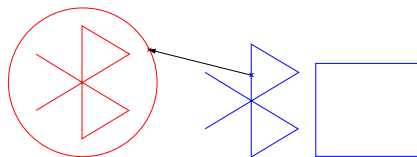
- Take random points $a \in A$ and $b \in B$ and insert vector $v = b - a$ into a sample of **translation space**
- Repeat this experiment to obtain an approximation of the probability distribution in translation space
- Possible candidates = centers of **clusters**



Probabilistic Matching

For matching A to B by translation:

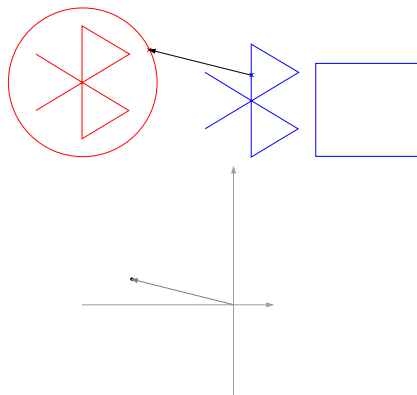
- Take random points $a \in A$ and $b \in B$ and insert vector $v = b - a$ into a sample of **translation space**
- Repeat this experiment to obtain an approximation of the probability distribution in translation space
- Possible candidates = centers of **clusters**



Probabilistic Matching

For matching A to B by translation:

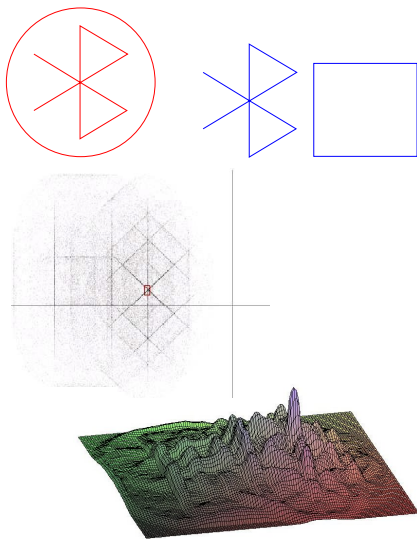
- Take random points $a \in A$ and $b \in B$ and insert vector $v = b - a$ into a sample of **translation space**
- Repeat this experiment to obtain an approximation of the probability distribution in translation space
- Possible candidates = centers of **clusters**



Probabilistic Matching

For matching A to B by translation:

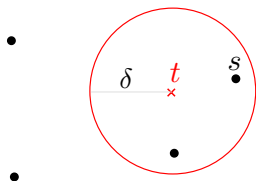
- Take random points $a \in A$ and $b \in B$ and insert vector $v = b - a$ into a sample of **translation space**
- Repeat this experiment to obtain an approximation of the probability distribution in translation space
- Possible candidates = centers of **clusters**



Finding Clusters

Goal: For a given cluster size δ find a translation having most sampling translation vectors in its δ -neighborhood.

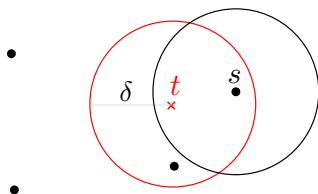
Observation: if translation vector t has sampling vector s in its δ -neighborhood, then t is contained in δ -neighborhood of s .



Finding Clusters

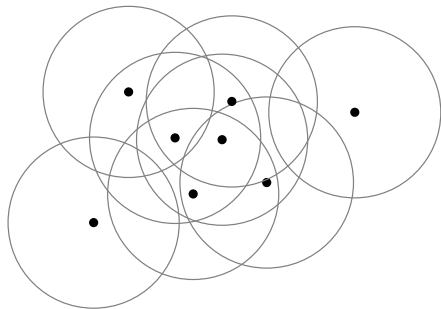
Goal: For a given cluster size δ find a translation having most sampling translation vectors in its δ -neighborhood.

Observation: if translation vector t has sampling vector s in its δ -neighborhood, then t is contained in δ -neighborhood of s .



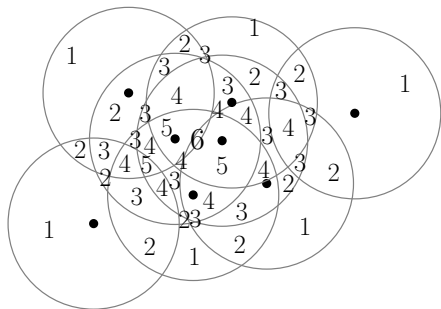
Finding Clusteres

- Consider arrangement of δ -neighborhoods of the sampling set.
- For every cell of the arrangement record the number of δ -neighborhoods it is covered by.
- Take the cells with the highest coverage.



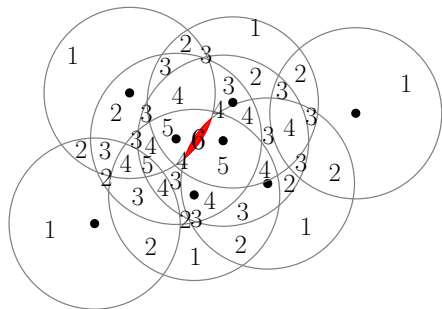
Finding Clusteres

- Consider arrangement of δ -neighborhoods of the sampling set.
- For every cell of the arrangement record the number of δ -neighborhoods it is covered by.
- Take the cells with the highest coverage.

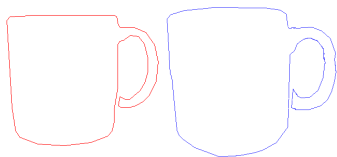


Finding Clusteres

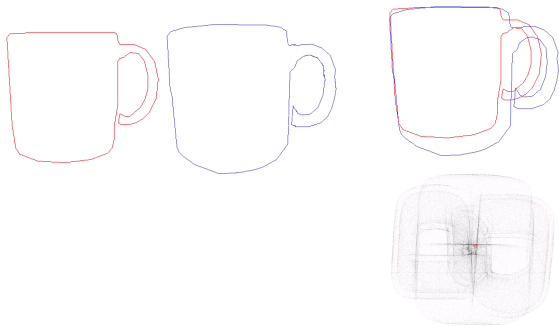
- Consider arrangement of δ -neighborhoods of the sampling set.
- For every cell of the arrangement record the number of δ -neighborhoods it is covered by.
- Take the cells with the highest coverage.



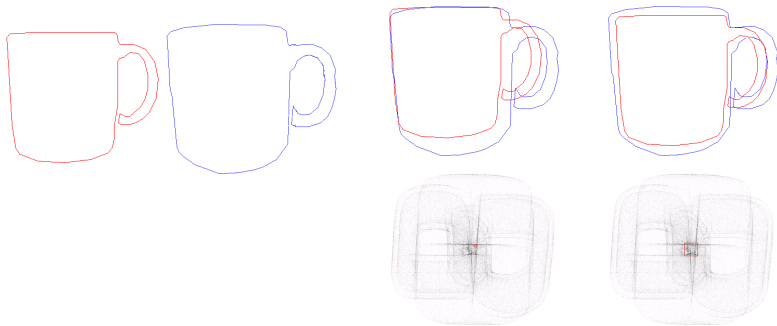
Role of δ



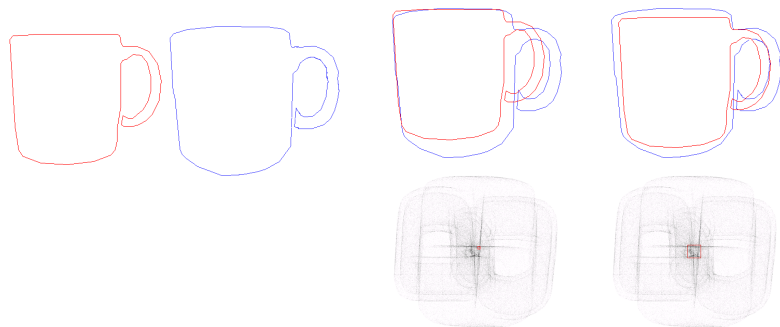
Role of δ



Role of δ



Role of δ



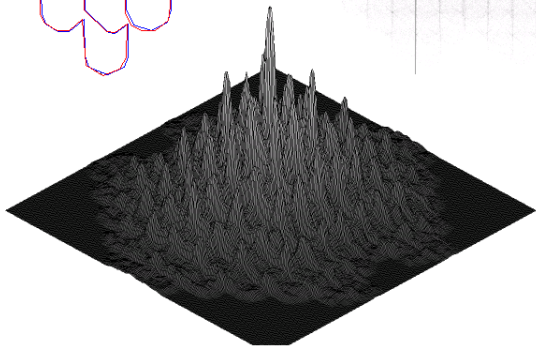
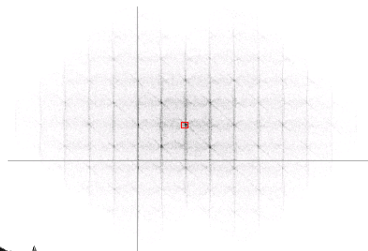
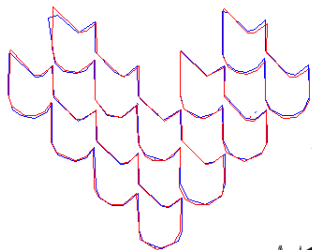
- Point t_{max} maximizes the **(Lebesgue measure)** of the set $M(t) = \{(a, b) : a \in A, b \in B, \|a - t(b)\| \leq \delta\}$

Running time

- similarity function $f(t) = |M(t)|/|A \times B|$
- With $N \in O\left(\log \frac{1}{\eta} / \varepsilon^2\right)$ samples we get an approximation of maximum of $f(t)$ with error at most ε with probability at least $1 - \eta$. (Idea from [Cheong/Efrat/Har-Peled 2004](#))
- Generating N random points takes $O(n + N \log n)$ time.
- Arrangement construction and traversal in $O(N^2)$.
- Total complexity is $O(n + N \log n + N^2)$.

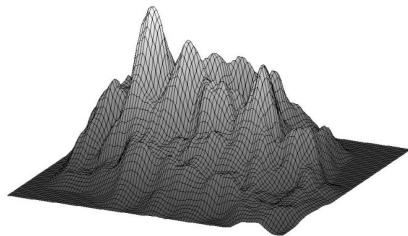
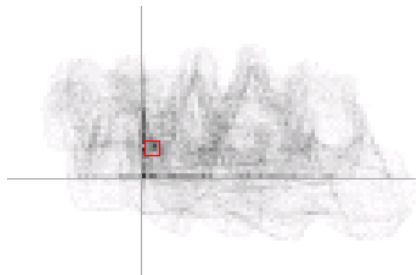
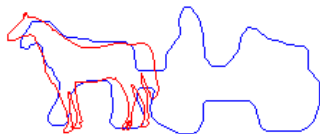
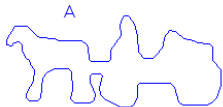
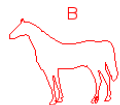
Examples

Complete-Complete Matching



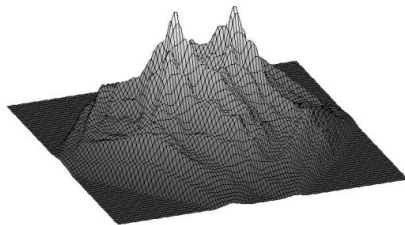
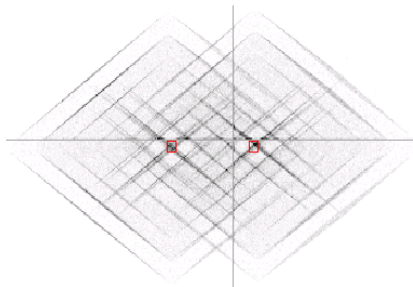
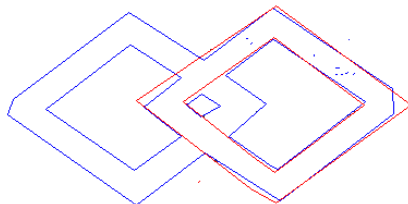
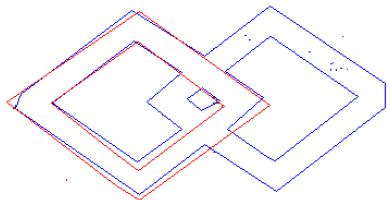
Examples

Complete-Partial Matching I



Examples

Complete-Partial Matching II



A faster Heuristic

Idea: take samples of only “promising” transformations

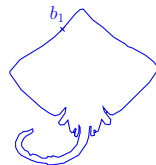
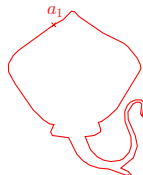
- use vertices as features and the structure defined by the polylines
- give weight to every sample transformation
- collect samples into clusters
- the biggest clusters provide the candidate transformations

Computing a Vote



Computing a Vote

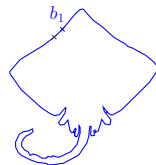
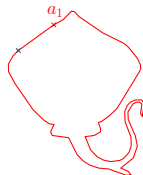
- choose randomly vertices $a_1 \in A$ and $b_1 \in B$ and add (a_1, b_1) to the sample set S



Computing a Vote

- choose randomly vertices $a_1 \in A$ and $b_1 \in B$ and add (a_1, b_1) to the sample set S
- extend sample as long as the error is small enough:

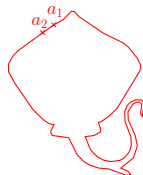
$$\varepsilon(t) = \sum_{(a_i, b_i) \in S} w(a_i, b_i) \|a_i - t(b_i)\|^2$$



Computing a Vote

- choose randomly vertices $a_1 \in A$ and $b_1 \in B$ and add (a_1, b_1) to the sample set S
- extend sample as long as the error is small enough:

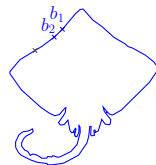
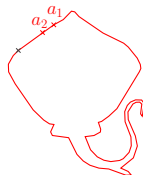
$$\varepsilon(t) = \sum_{(a_i, b_i) \in S} w(a_i, b_i) \|a_i - t(b_i)\|^2$$



Computing a Vote

- choose randomly vertices $a_1 \in A$ and $b_1 \in B$ and add (a_1, b_1) to the sample set S
- extend sample as long as the error is small enough:

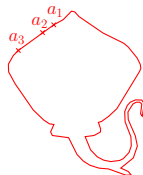
$$\varepsilon(t) = \sum_{(a_i, b_i) \in S} w(a_i, b_i) \|a_i - t(b_i)\|^2$$



Computing a Vote

- choose randomly vertices $a_1 \in A$ and $b_1 \in B$ and add (a_1, b_1) to the sample set S
- extend sample as long as the error is small enough:

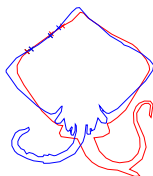
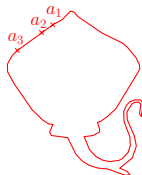
$$\varepsilon(t) = \sum_{(a_i, b_i) \in S} w(a_i, b_i) \|a_i - t(b_i)\|^2$$



Computing a Vote

- choose randomly vertices $a_1 \in A$ and $b_1 \in B$ and add (a_1, b_1) to the sample set S
- extend sample as long as the error is small enough:

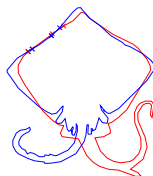
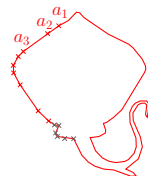
$$\varepsilon(t) = \sum_{(a_i, b_i) \in S} w(a_i, b_i) \|a_i - t(b_i)\|^2$$



Computing a Vote

- choose randomly vertices $a_1 \in A$ and $b_1 \in B$ and add (a_1, b_1) to the sample set S
- extend sample as long as the error is small enough:

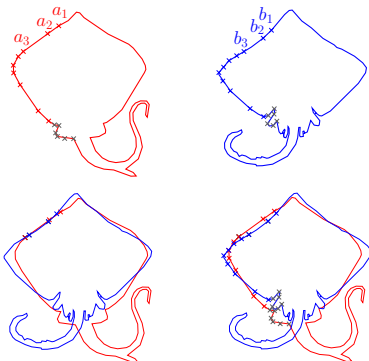
$$\varepsilon(t) = \sum_{(a_i, b_i) \in S} w(a_i, b_i) \|a_i - t(b_i)\|^2$$



Computing a Vote

- choose randomly vertices $a_1 \in A$ and $b_1 \in B$ and add (a_1, b_1) to the sample set S
- extend sample as long as the error is small enough:

$$\varepsilon(t) = \sum_{(a_i, b_i) \in S} w(a_i, b_i) \|a_i - t(b_i)\|^2$$

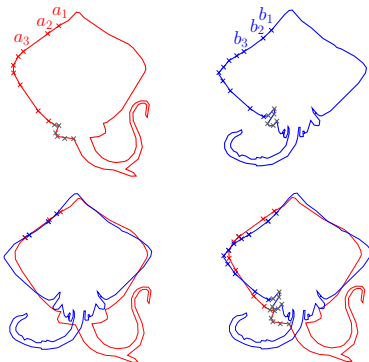


Computing a Vote

- choose randomly vertices $a_1 \in A$ and $b_1 \in B$ and add (a_1, b_1) to the sample set S
- extend sample as long as the error is small enough:

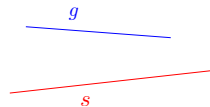
$$\varepsilon(t) = \sum_{(a_i, b_i) \in S} w(a_i, b_i) \|a_i - t(b_i)\|^2$$

- compute weight of the resulting transformation



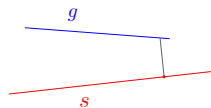
Similarity Function

- distance



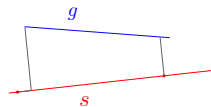
Similarity Function

- distance $\delta_{s,g}(t)$



Similarity Function

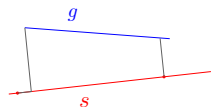
- distance $\delta_{s,g}(t)$



Similarity Function

- inverse distance

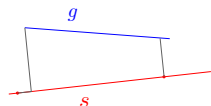
$$\alpha_{s,g}(t) = \max \left\{ 1 - 25 \left(\frac{\delta_{s,g}(t)}{D_A} \right)^2, 0 \right\}$$



Similarity Function

- inverse distance

$$\alpha_{s,g}(t) = \max \left\{ 1 - 25 \left(\frac{\delta_{s,g}(t)}{D_A} \right)^2, 0 \right\}$$

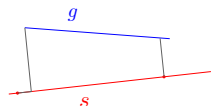


- slope $\beta_{s,g} = \cos^4(\angle(s, g))$

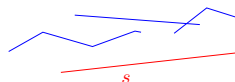
Similarity Function

- inverse distance

$$\alpha_{s,g}(t) = \max \left\{ 1 - 25 \left(\frac{\delta_{s,g}(t)}{D_A} \right)^2, 0 \right\}$$



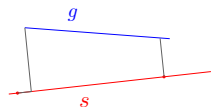
- slope $\beta_{s,g} = \cos^4(\angle(s, g))$
- combined: $\phi_s(t) = \max_{g \in B} (\alpha_{s,g}(t) \cdot \beta_{s,g})$



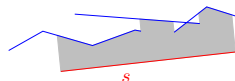
Similarity Function

- inverse distance

$$\alpha_{s,g}(t) = \max \left\{ 1 - 25 \left(\frac{\delta_{s,g}(t)}{D_A} \right)^2, 0 \right\}$$



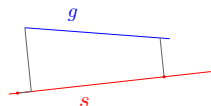
- slope $\beta_{s,g} = \cos^4(\angle(s, g))$
- combined: $\phi_s(t) = \max_{g \in B} (\alpha_{s,g}(t) \cdot \beta_{s,g})$



Similarity Function

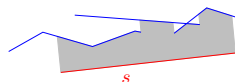
- inverse distance

$$\alpha_{s,g}(t) = \max \left\{ 1 - 25 \left(\frac{\delta_{s,g}(t)}{D_A} \right)^2, 0 \right\}$$



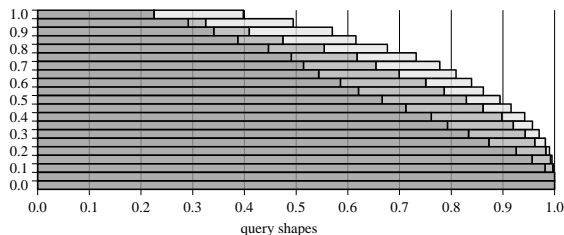
- slope $\beta_{s,g} = \cos^4(\angle(s, g))$
- combined: $\phi_s(t) = \max_{g \in B} (\alpha_{s,g}(t) \cdot \beta_{s,g})$
- directed similarity:

$$\Phi_{\rightarrow}(A, B) = \frac{\sum_{s \in A} \left(\int_{t=0}^1 \phi_s(t) \cdot w_s(t) dt \cdot l_s \right)}{W(A)}$$



Experimental results

MPEG-7 shape B set: 1400 images of shapes, 70 classes, each class contains 20 shapes.



Average ratio of shapes from the same class found

- as consecutive first nearest neighbors: 64.74%
- among the 20 nearest neighbors: 74.40%
- among the 40 nearest neighbors: 82.09%

Experimental results

Query Shape	True Positives		
	as Nearest Neighbours	in Class Size	in Double the Class Size
apple	89%	97%	100%
bat	86%	90%	96%
beetle	59%	70%	75%
bell	95%	96%	100%
bird	51%	61%	65%
Bone	57%	73%	86%
bottle	94%	97%	100%
brick	100%	100%	100%
butterfly	65%	71%	81%
camel	73%	86%	92%
car	99%	99%	100%
carriage	100%	100%	100%
cattle	54%	63%	64%
cellular_phone	94%	96%	98%
chicken	38%	58%	75%
children	100%	100%	100%
chopper	98%	98%	100%
classic	64%	72%	82%
Comma	81%	87%	94%
crown	80%	85%	94%
cup	81%	86%	92%
deer	37%	43%	47%
device0	88%	92%	98%
device1	67%	73%	76%
device2	44%	52%	55%
device3	40%	57%	70%
device4	52%	68%	82%
device5	91%	95%	99%
device6	43%	63%	72%
device7	98%	98%	100%
device8	72%	80%	89%
device9	22%	42%	55%
dog	20%	42%	61%
elephant	46%	69%	82%
face	100%	100%	100%

Query Shape	True Positives		
	as Nearest Neighbours	in Class Size	in Double the Class Size
fish	32%	53%	68%
flatfish	88%	92%	99%
fly	31%	48%	64%
fork	68%	78%	93%
fountain	100%	100%	100%
frog	40%	49%	57%
Glas	100%	100%	100%
guitar	40%	60%	79%
hammer	80%	86%	92%
hat	24%	35%	44%
HCircle	78%	87%	97%
Heart	93%	95%	100%
horse	35%	53%	64%
horseshoe	74%	87%	95%
jar	44%	58%	72%
key	50%	71%	88%
lizzard	16%	28%	32%
lmfish	50%	78%	93%
Misk	79%	85%	91%
octopus	18%	28%	33%
pencil	71%	85%	98%
personal_car	75%	86%	93%
pocket	64%	66%	73%
rat	100%	100%	100%
ray	42%	76%	92%
sea_snake	52%	65%	74%
shoe	87%	88%	97%
spoon	21%	34%	46%
spring	53%	62%	72%
stef	35%	55%	66%
teddy	100%	100%	100%
tree	39%	44%	48%
truck	100%	100%	100%
turtle	31%	47%	56%
watch	56%	77%	95%