Unsupervised Image Segmentation Using Comparative Reasoning and Random Walks

Anuva Kulkarni Carnegie Mellon University
Filipe Condessa Carnegie Mellon, IST-Lisbon
Jelena Kovacevic Carnegie Mellon University

Outline

- Motivation
  - Training-free methods
  - Hashing
  - Related work
- Approach
  - Winner Take All (WTA) Hash
  - Clustering based on Random Walks
- Some experimental results

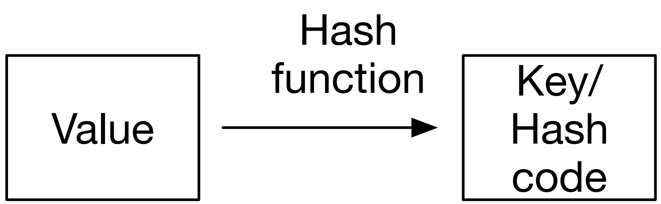
Motivation

- Goals:
  - Segment images where no. of classes unknown)

  - Eliminate training-data (may not be available)
  - Fast computation as a pre-processing step for classification
- Segmentation is similarity-search
- Machine learning concept of “hashing” data for fast similarity-search

Hashing

- Used to speed up the searching process
- A ‘hash function’ relates the data values to keys or ‘hash codes’
- Hash table : shortened representation of data

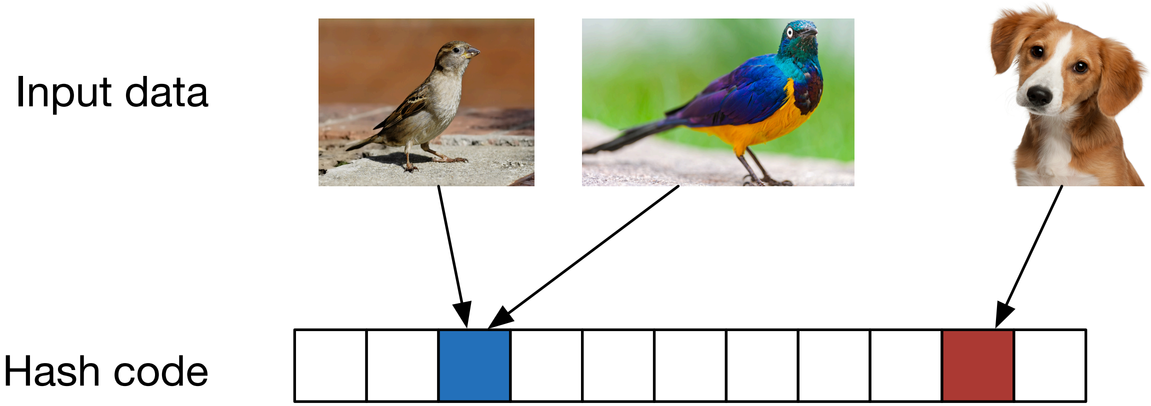


Hash table

Hash value	Data
001	Bird_type1
010	Bird_type2
011	Dog_type1
100	Fox_type1

Hashing

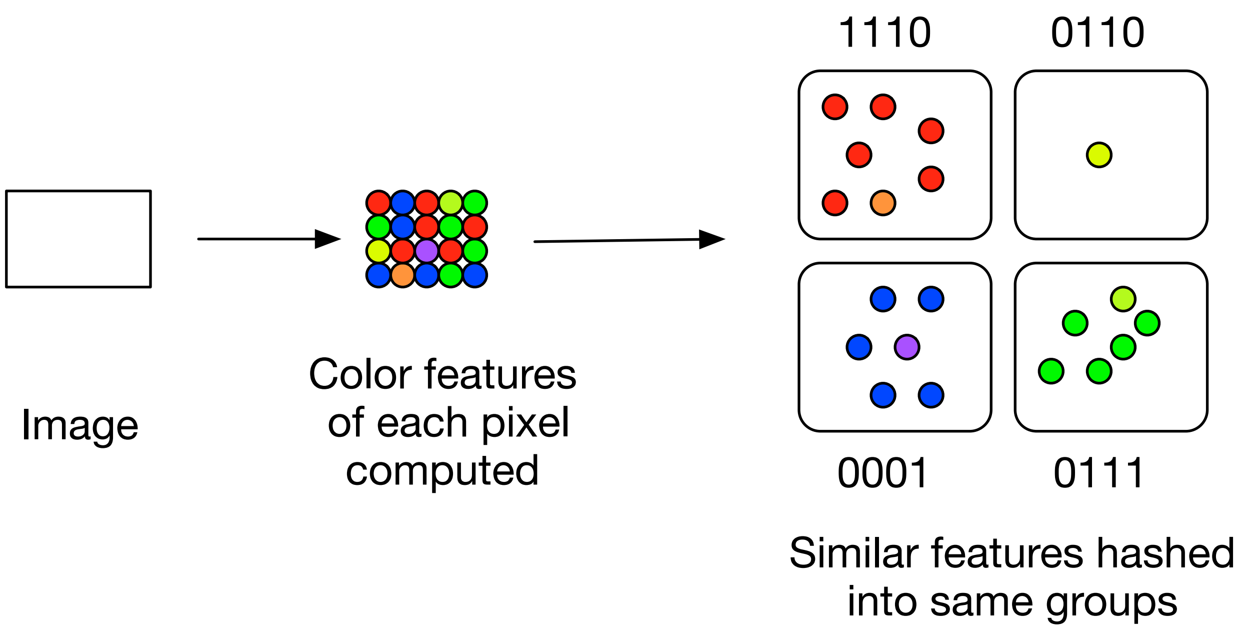
- Similar data points have the same (or close by) hash values



- Hash function:
  - Always returns a number for an object
  - Two equal objects will always have the same number
  - Two unequal objects may not always have different numbers

Hashing for Segmentation

- Each pixel is described by some feature vectors (eg. Color)
- Hashing is used to cluster them into groups



Segmentation and Randomized Hashing

- Used by Taylor and Cowley (2009) for image segmentation

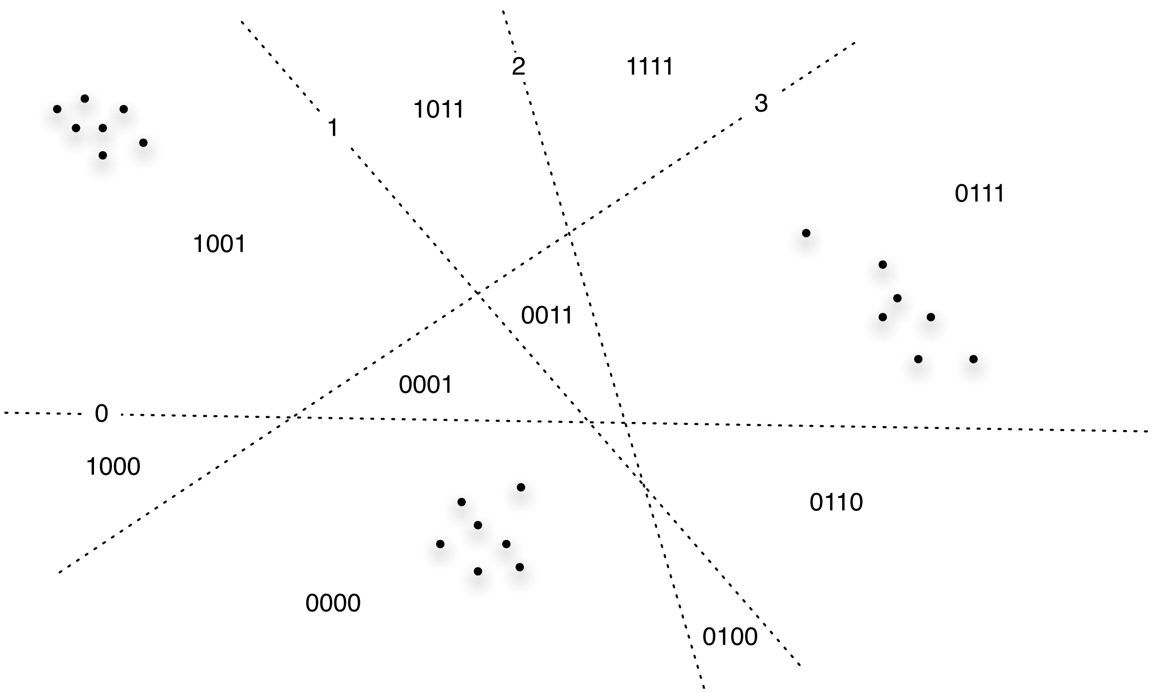
- Algorithm:
  - Hash the features of each pixel into *n*-bit codes
  - Find local maxima in the space of hash codes. These are ”cluster centers”
  - Assign feature vector to closest maxima → get clusters
  - Use a connected components algorithm

- Parallelizable

C. J. Taylor and A. Cowley, “Fast segmentation via randomized hashing.,” in BMVC, pp. 1–11, 2009.

Segmentation and Randomized Hashing

- Random hashing i.e using a hash code to indicate the region in which a feature vector lies after splitting the space using a set of randomly chosen splitting planes



C. J. Taylor and A. Cowley, “Fast segmentation via randomized hashing.,” in BMVC, pp. 1–11, 2009.

Winner Take All Hash

- A way to convert feature vectors into compact binary hash codes
- Rank correlation is preserved
- Absolute value of feature does not matter; only the ordering of values matters

- Distance between hashes approximates rank correlation (?)

C. J. Taylor and A. Cowley, “Fast segmentation via randomized hashing.,” in BMVC, pp. 1–11, 2009.

Calculating WTA Hash

- Consider 3 feature vectors

- Step 1: Create random permutations

Permutation vector θ

3	1	5	2	6	4

	feature 1						feature 2						feature 3						
	13	4	2	11	5	3	12	5	3	10	4	2	1	90	44	5	15	6	
Step 1	2	13	5	4	3	11	3	12	4	5	2	10	44	1	15	90	6	5	Permute with θ

Calculating WTA Hash

- Step 2: Choose first K entries. Let K=3

Permutation vector θ

3	1	5	2	6	4

	feature 1						feature 2						feature 3						
	13	4	2	11	5	3	12	5	3	10	4	2	1	90	44	5	15	6	
Step 1	2	13	5	4	3	11	3	12	4	5	2	10	44	1	15	90	6	5	Permute with θ
Step 2	2	13	5	4	3	11	3	12	4	5	2	10	44	1	15	90	6	5	Choose first K entries

Calculating WTA Hash

- Step 3: Pick the index of the max. entry. This is the hash code 'h' of that feature vector

Permutation vector θ						
	3	1	5	2	6	4

	feature 1						feature 2						feature 3						
	13	4	2	11	5	3	12	5	3	10	4	2	1	90	44	5	15	6	
Step 1	2	13	5	4	3	11	3	12	4	5	2	10	44	1	15	90	6	5	Permute with θ
Step 2	2	13	5	4	3	11	3	12	4	5	2	10	44	1	15	90	6	5	Choose first K entries
Step 3	2	13	5	4	3	11	3	12	4	5	2	10	44	1	15	90	6	5	Hash code is index of top entry out of the K
	h=2						h=2						h=1						

Calculating WTA Hash

Notice that Feature 2 is just Feature 1 perturbed by one, but Feature 3 is very different

Permutation vector θ

3	1	5	2	6	4

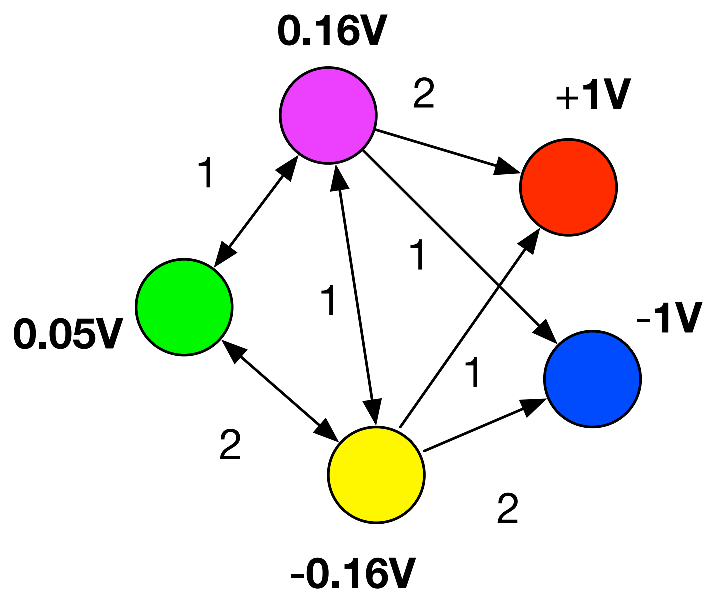
	feature 1	feature 2	feature 3	
	13 4 2 11 5 3	12 5 3 10 4 2	1 90 44 5 15 6	
Step 1	2 13 5 4 3 11	3 12 4 5 2 10	44 1 15 90 6 5	Permute with θ
Step 2	2 13 5 4 3 11	3 12 4 5 2 10	44 1 15 90 6 5	Choose first K entries
Step 3	2 13 5 4 3 11	3 12 4 5 2 10	44 1 15 90 6 5	Hash code is index of top entry out of the K
	h=2	h=2	h=1	

Feature 1 and Feature 2 are similar

Random Walks

- Understanding proximity in graphs
- Useful in **propagation** in graphs

- Similar to electrical network with voltages and edge weights inversely proportional to resistances



Calculating WTA Hash

- Consider a feature vector

12	5	1	33	7	15

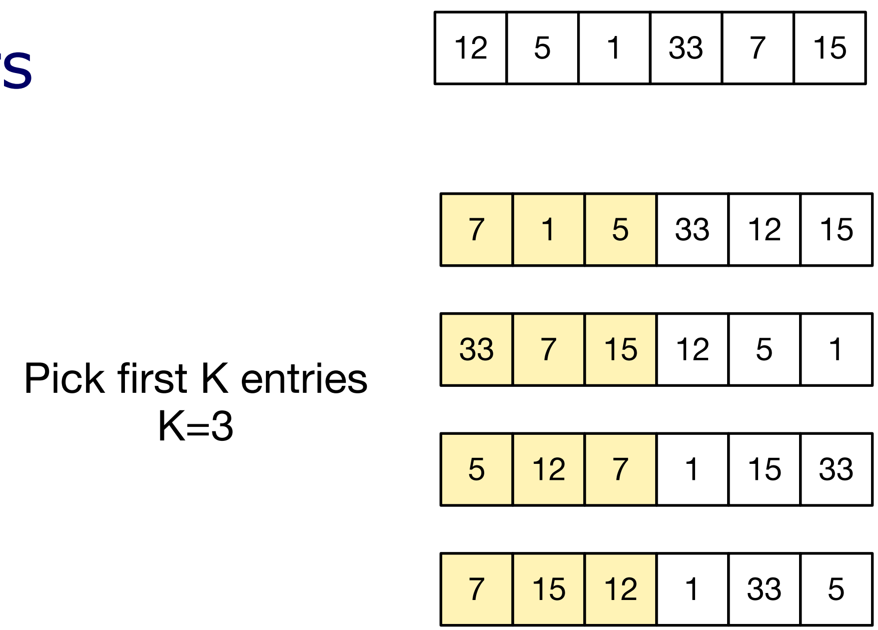
- Step 1: Create P=4 random permutations

4 random permutations

7	1	5	33	12	15
33	7	15	12	5	1
5	12	7	1	15	33
7	15	12	1	33	5

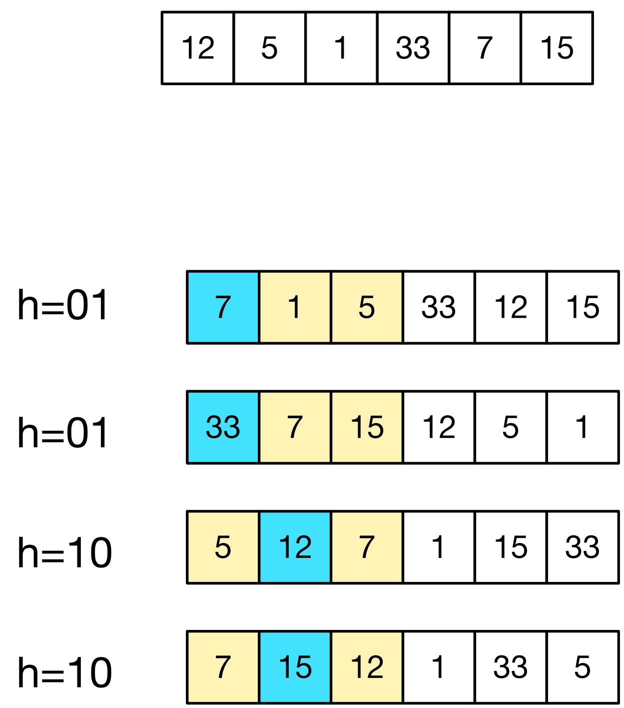
Calculating WTA Hash

- Step 2: Pick first K entries of the permuted vectors
- K=3



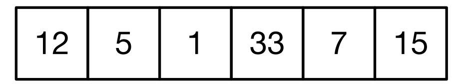
Calculating WTA Hash

- Step 3: Index of the maximum element is the hash code
- Thus a binary code is associated with our feature vector

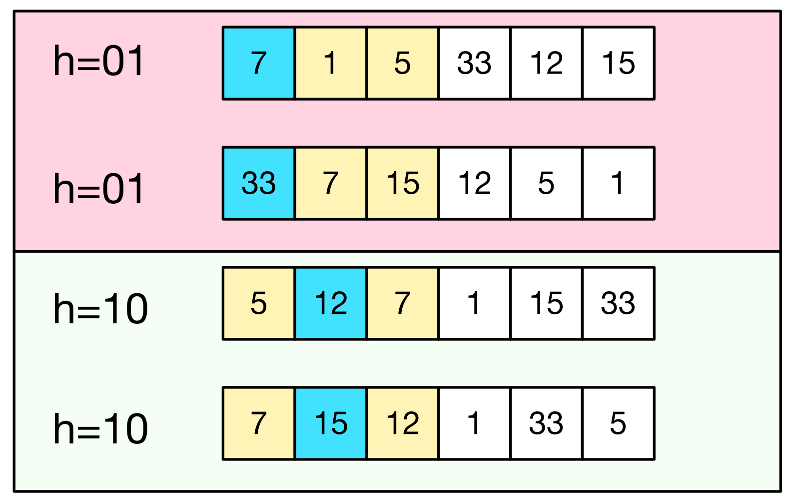


Calculating WTA Hash

- Step 4: Bin features according to the similarity in their hash codes

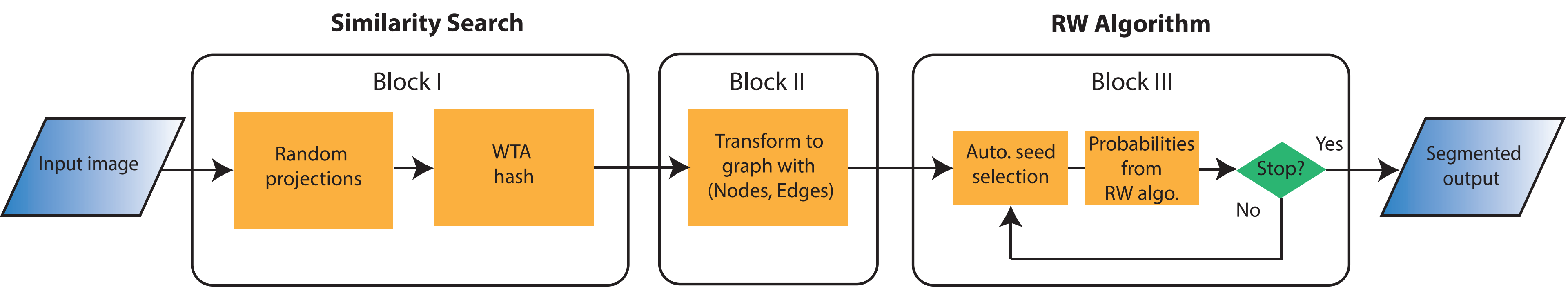


- MinHash is a special case of WTA Hash



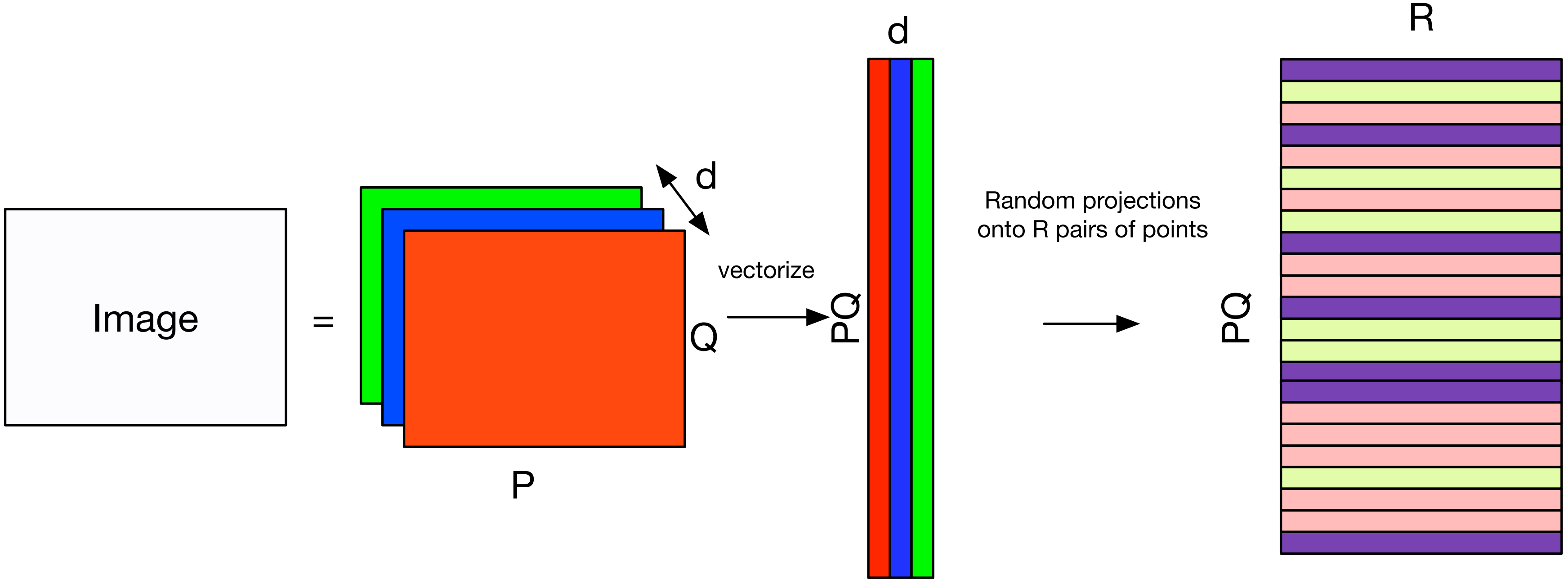
Our Approach

1. Similarity Search using WTA Hash
2. Transformation to graph with nodes and edges
3. Probability map using Random Walks
   – Automatic seed selection
4. Clustering



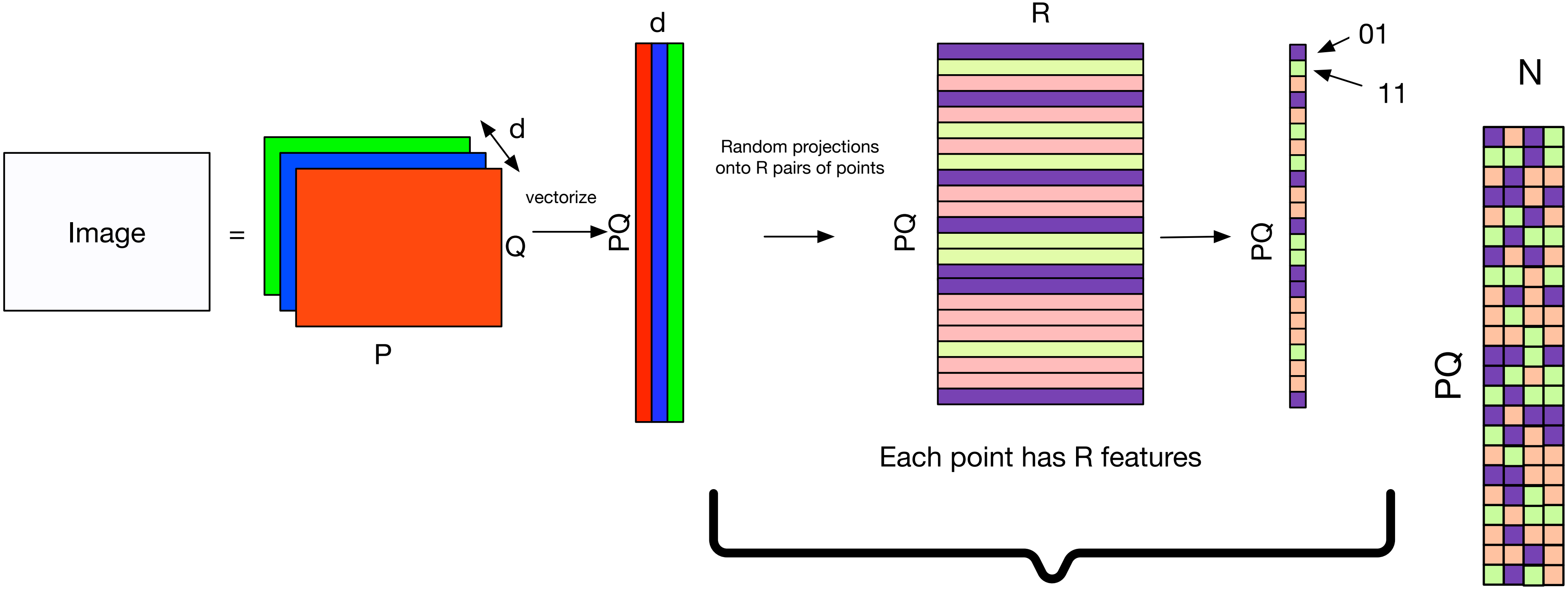
Block I: WTA hash

- Image Dimensions: P x Q x d
- Project onto R randomly chosen hyperplanes
  - Each point in image has R feature vectors



Block I: WTA hash

- Run WTA hash *N* times.



K=3
Hence possible values
of hash codes
are 00, 01, 11

Repeat this N times to get PQ x N matrix of hash codes

Block II: Create Graph

- Run WTA hash N times → each point has N hash codes
- Image transformed into lattice
- Edge weights: $w_{i,j} = \exp(-\beta v_{i,j})$

Where:

$$v_{i,j} = \frac{d_H(i,j)}{\gamma}$$

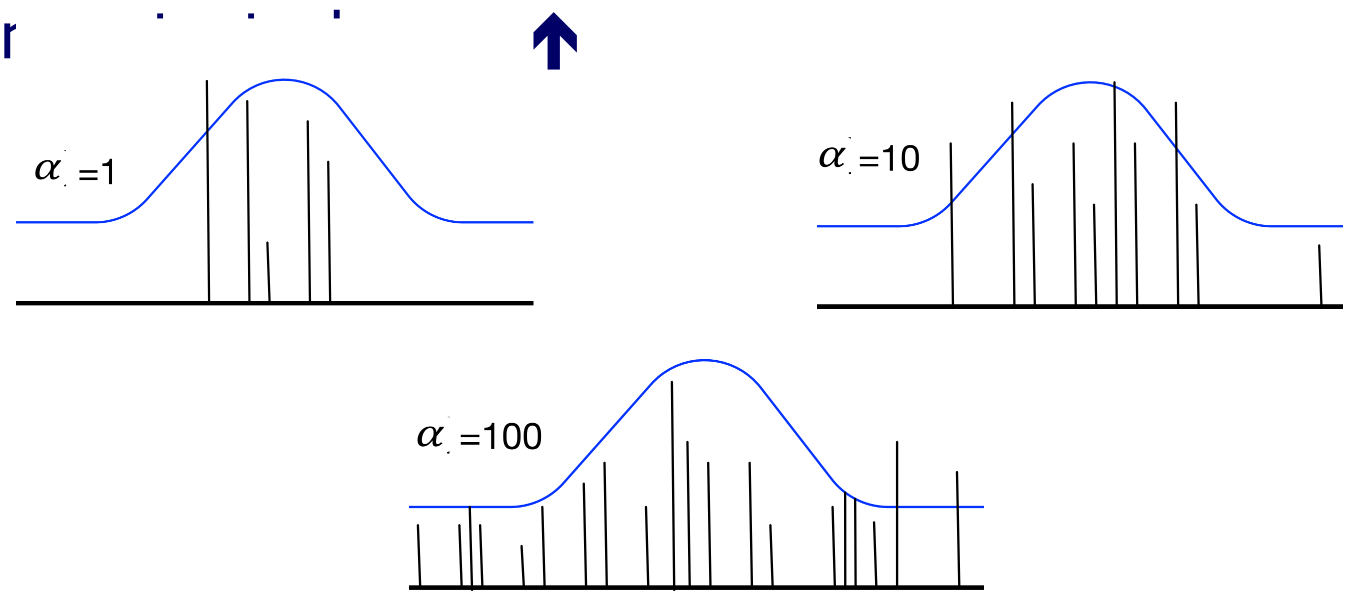
$d_H(i,j) =$ avg. Hamming distance over all N hash codes of nodes i and j

$\gamma =$ Scaling factor

$\beta =$ Weight parameter for RW algorithm

Block III: Random Walks

- Needs initial seeds to be defined
- Unsupervised draws using Dirichlet processes
- DP(G_0,α)
  - G_o is base distribution
  - α is concentration parameter
- DP draws values around G_0. Samples are less cor



Block III: Random Walks

- Draw seeds from Dirichlet process DP(G,α) with base distribution G_0
- $X_1, \ldots X_{n-1}$ are samples drawn from the Dirichlet process
- Behaviour of the next sample X_n given the previous samples is:

$$X_n \mid X_1, \ldots X_{n-1} = \begin{cases} X_i \text{ with prob. } \dfrac{1}{n-1+\alpha} \\ \text{New draw from } \mathrm{G}_0 \text{ with prob. } \dfrac{\alpha}{n-1+\alpha} \end{cases}$$

Block III: Random Walks

- Probability that a new seed belongs to a new class is proportional to α
- Posterior probability for the *i*th sample with class label y_i :

$$p(y_i = c|\boldsymbol{y}_{-i}, \alpha) = \frac{n_c^{-i} + \frac{\alpha}{C_{tot}}}{n-1+\alpha}$$

where
C_{tot} = Total number of classes
y_i = Class label $c, c \in \{1,2 \dots C_{tot}\}$
$\boldsymbol{y}_{-i} = \{y_j | \, j \neq \, i\}$
n_c^{-i} = number of samples in cth class excluding the ith sample

Block III: Random Walks

- Unsupervised, hence C_{tot} is infinite. Hence,

$$\lim_{C_{tot}\to\infty} p(y_i = c|\boldsymbol{y}_{-i}, \alpha) = \frac{n_c^{-i}}{n-1+\alpha}, \qquad \forall c, n_c^{-i} > 0$$

Class is non-empty

- "Clustering effect" or "rich gets richer"

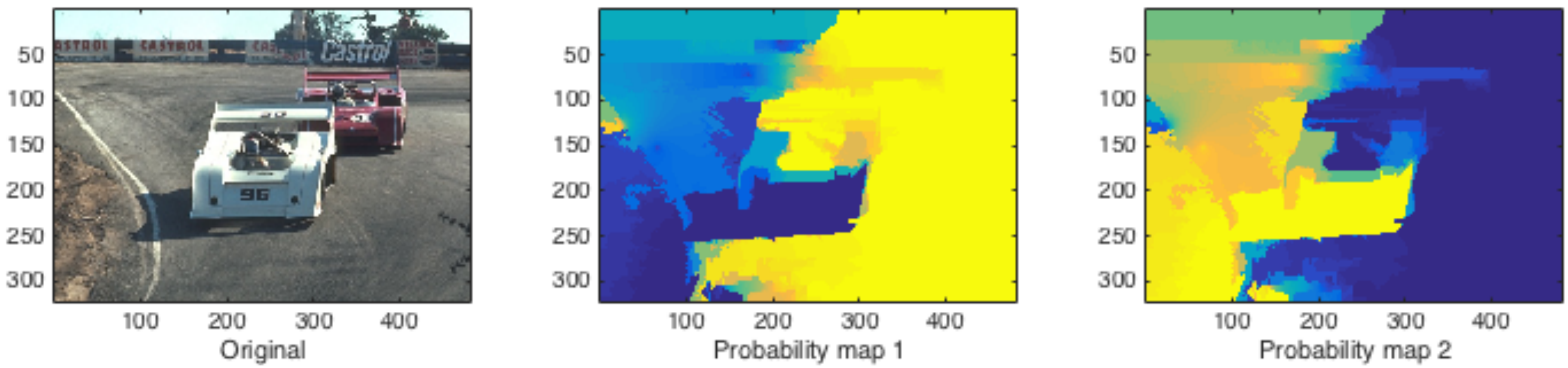
- Probability that a new class is discovered:

$$\lim_{C_{tot}\to\infty} \sum_c p(y_i = c|\boldsymbol{y}_{-i}, \alpha) = \frac{\alpha}{n-1+\alpha}, \qquad \forall c,\ n_c^{-i} = 0$$

Class is empty or new

Block III: Random Walks

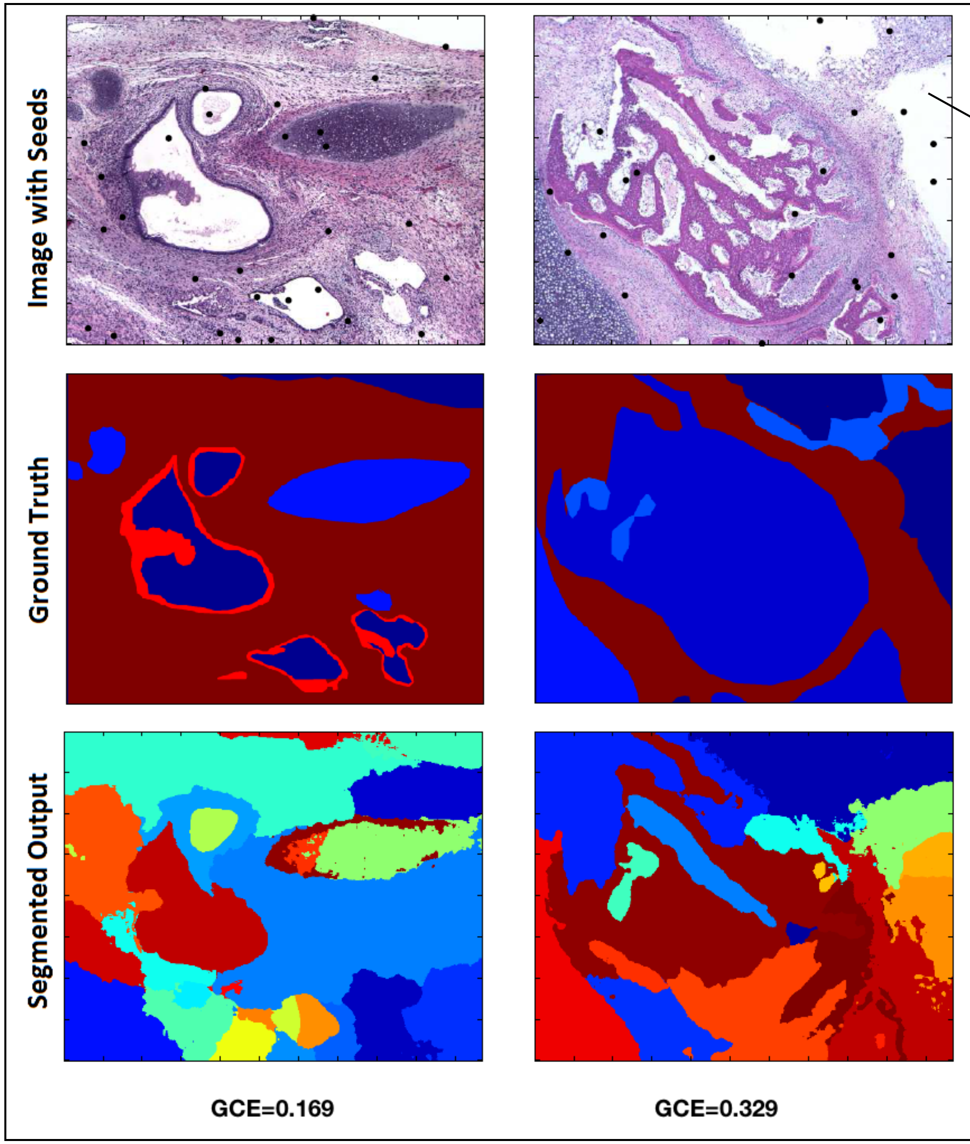
- Use the RW algorithm to generate *c* probability maps, *c*= Number of classes found so far.



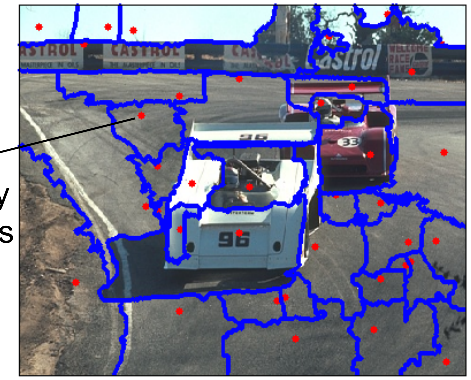
- Entropy calculated with probability maps
- Entropy-based stopping criteria
  - Cluster purity ↑, Avg. image entropy ↓

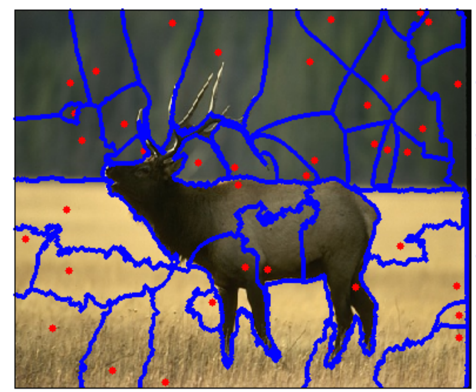
Experimental Results

Histology images



Automatically
Picked seeds

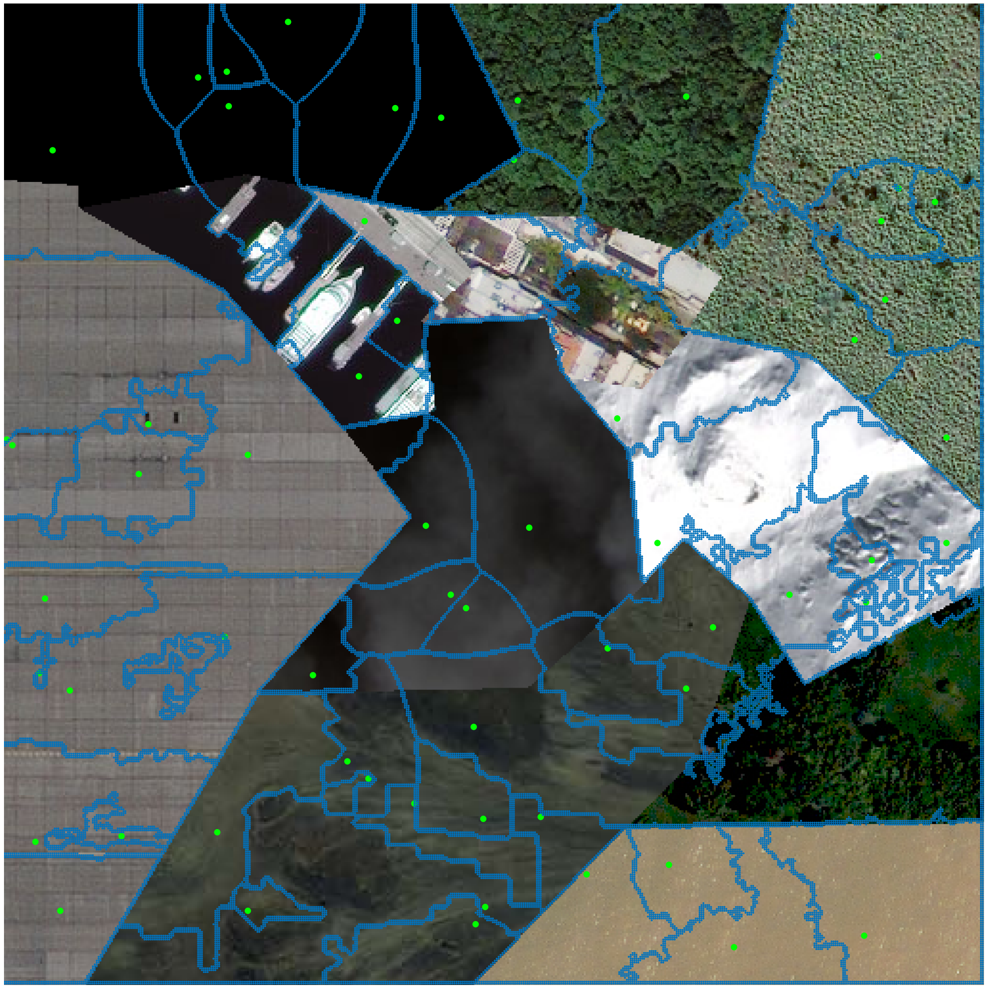




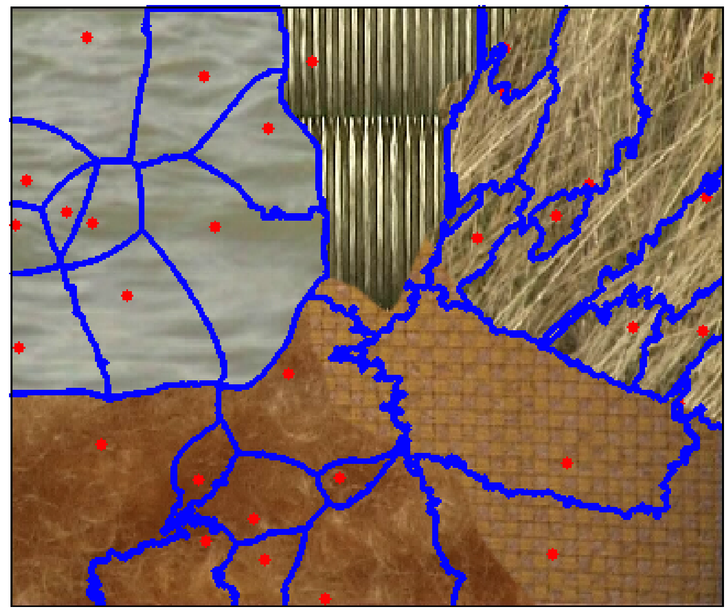
Berkeley segmentation subset
Avg. accuracy = 91.42% ± 4.57

Experimental Results

TexGeo
Avg. accuracy = 95.14% ± 2.97



TexBTF
Avg. accuracy= 98.36% ± 0.78



Experimental Results

- Comparison measure: Global Consistency Error (GCE)*
  - Lower GCE indicates lower error

Value Of R	GCE Score			
	BSDSubset	TexBTF	TexColor	TexGeo
10	0.179	0.063	0.159	0.102
20	0.180	0.065	0.159	0.129
40	0.186	0.061	0.156	0.134

*C. Fowlkes, D. Martin, and J. Malik, "Learning affinity functions for image segmentation: Combining patch-based and gradient-based approaches," vol. 2, pp. II–54, IEEE, 2003.

Experimental Results

- Comparison measure: Global Consistency Error (GCE)
  - Lower GCE indicates lower error

Value Of R	GCE Score			
	BSDSubset	TexBTF	TexColor	TexGeo
10	0.179	0.063	0.159	0.102
20	0.180	0.065	0.159	0.129
40	0.186	0.061	0.156	0.134

- Comparison with other methods**:

Method	Human	RAD	Seed	Learned Affinity	Mean Shift	Normalized cuts
GCE	**0.080**	0.205	0.209	0.214	0.260	0.336

**E. Vazquez, J. Van De Weijer, and R. Baldrich, “Image segmentation in the presence of shadows and highlights,” pp. 1–14, Springer, 2008.

Conclusion

- WTA enables fast similarity search
- Parallelizable
- Completely unsupervised Random Walks-based clustering
- Can be used as pre-processing step in classification for images where number of classes is unknown