Code: https://github.com/layer6ai-labs/EGT

Explore-Exploit Graph Traversal for Image Retrieval

Cheng Chang*, Guangwei Yu*, Chundi Liu, Maksims Volkovs.
Layer6 AI

CVPR LONG BEACH CALIFORNIA June 16-20, 2019

Introduction

- Given a query image, retrieve its most relevant images from an index.

Figure 1: Example query image (left) and four retrieved results (right). Green indicate relevant match while red indicate irrelevant match.

- **Motivation**: Relevant images may be visually dissimilar to query, but similar to another image that *is* similar to query.
- **Main Idea**: Traverse the k-nearest neighbor (k-NN) graph to build "trusted" paths between images.

Figure 2: Example retrieval that illustrate our idea. Visually dissimilar image (second in green) is retrieved based on its visually similar neighbors.

k-NN Graph

- Following existing work, we start by building a k-NN graph on the index images.
- Here, vertices correspond to images and edges are weighted by a predefined similarity function.
- We construct the k-NN graph using global descriptor retrieval, and set edge weights to be the dot product between descriptors.

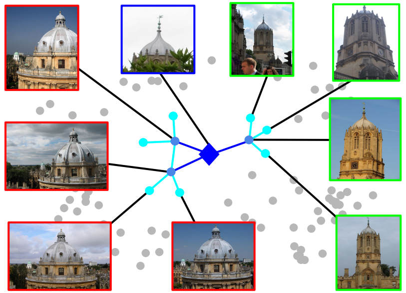

Figure 3: Example k-NN graph with k = 3. Blue, green, and red denote query, relevant and irrelevant images respectively. Shades of blue denote the neighbors.

Explore-Exploit Graph Traversal (EGT)

- We traverse the k-NN graph starting from query and incrementally build "trusted" paths between vertices by alternating between explore and exploit steps.
- **Explore**: neighbors of trusted vertices get added to the explore priority queue ordered by edge weight.
- **Exploit**: vertices with edge weights that pass a given threshold get retrieved from the queue and become trusted.
- Edge weight threshold *t* plays a critical role in our algorithm and controls the degree of exploration.

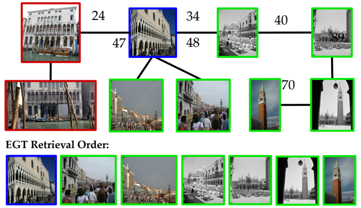

Figure 4: Example rEGT graph on Google Landmark'18. Blue, green and red denote query, relevant, and irrelevant images.

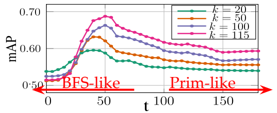

Figure 5: Effect of t on ROxford: at t=0 EGT is analogous to *breadth-first search (BFS)*, while $t \to \infty$ is analogous to *Prim's algorithm*.

Algorithm 1: EGT

input : k-NN graph $G_k = (\mathcal{X}, A_k, s_k)$,
query u,
number of images to retrieve p,
edge weight threshold t
output: list of retrieved images Q

initialize max-heap H, list V, and list Q
add u to V
do
  // Explore step
  foreach $v \in V$ **do**
    foreach $x \in NN_k(v), x \notin Q, x \neq u$ **do**
      if $x \in H$ **and** $H[x] < s_k(v, x)$ **then**
        update weight for x: $H[x] \leftarrow s_k(v, x)$
      else if $x \notin H$ **then**
        push x to H with weight $s_k(v, x)$
      end
    end
  end
  clear V
  // Exploit step
  do
    $v \leftarrow pop(H)$
    add v to V and Q
  while $(peek(H) > t$ **or** $|V| = 0)$ **and** $|Q| < p$
while $|Q| < p$ **and** $|H| > 0$
return Q

Edge Re-weighting with Spatial Verification

- k-NN graphs computed from global descriptors (e.g. R-MAC) can be noisy making edge weights unreliable.
- We address this problem by re-weighting edges with spatial verification (rEGT). This significantly reduces topic drift and improves performance.

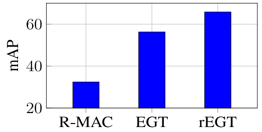

Figure 6: Performance comparison for rEGT on ROxford Hard.

Efficient Inference

- EGT is conceptually simple and can be efficiently implemented with standard data structures.
- Offline phase only requires k-NN graph construction.
- During online inference, EGT is greedy and independent of the index size, allowing it to scale to large databases.

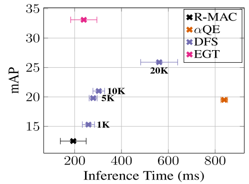

Figure 7: Performance vs. inference time on ROxford+R1M Hard

Experiments

Table1: Benchmark on ROxford and RParis with and without 1M distractor set, divided into with and without spatial verification (SV).

Method	mAP							
	$\mathcal{R}$Oxford		$\mathcal{R}$Oxford+$\mathcal{R}$1M		$\mathcal{R}$Paris		$\mathcal{R}$Paris+$\mathcal{R}$1M	
	Medium	Hard	Medium	Hard	Medium	Hard	Medium	Hard
Without SV								
R-MAC [11]	60.9	32.4	39.3	12.5	78.9	59.4	54.8	28.0
R-MAC+αQE [25]	64.8	36.8	45.7	19.5	82.7	65.7	61.0	35.0
R-MAC+DFS [15]	69.0	44.7	56.6	28.4	89.5	80.0	83.2	70.4
R-MAC+Hybrid-Spectral-Temporal [14]	67.0	44.2	55.6	27.2	89.3	80.2	82.9	69.2
R-MAC+EGT	73.6	56.3	55.8	35.1	90.6	81.2	79.4	63.7
With SV								
HesAff+rSIFT+HQE [29]+SV	71.3	49.7	52.0	29.8	70.2	45.1	46.8	21.8
DELF [20]+HQE +SV	73.4	50.3	60.6	37.9	84.0	69.3	65.2	35.8
HesAffNet+HardNet++ [19]+HQE+SV	75.2	53.3	-	-	73.1	48.9	-	-
R-MAC+DFS+DELF+ASMK [28]+SV	75.0	48.3	68.7	39.4	90.5	81.2	86.6	74.2
R-MAC+DFS+HesAff+rSIFT+ASMK+SV	80.2	54.8	**74.9**	47.5	92.5	84.0	**87.5**	**76.0**
R-MAC+QE+SV+rEGT	**83.5**	**65.8**	**74.9**	**54.1**	**92.8**	**84.6**	87.1	75.6

Figure 8: Top 5 results for diffusion (left) and rEGT (right) from selected queries in ROxford. Blue, green, and red denote query, relevant and irrelevant images respectively.