MO446 Computer Vision: Project 4

Ícaro Cavalcante Dourado
RA: 107396
Email: icaro.dourado@ic.unicamp.br

Juan Hernández
RA: 163128
Email: juan.albarracin@ic.unicamp.br

Miguel Rodriguez
RA: 192744
Email: m.rodriguezs1990@gmail.com

I. Introduction

This work consists of engineering a content-based image retrieval (CBIR) system. We adopt a solution based on Bag of Visual Words (BoVW), of image segment descriptors. We use and analyze two segment descriptor approaches. We validate our method over a supplied collection consisting of 40 images, distributed in 8 classes of 5 samples each, and 7 pre-defined queries. We validate our results using Precision and Average Precision. We achieved Precision at 3 and Average Precision at 3 of 80.95% in the supplied collection.

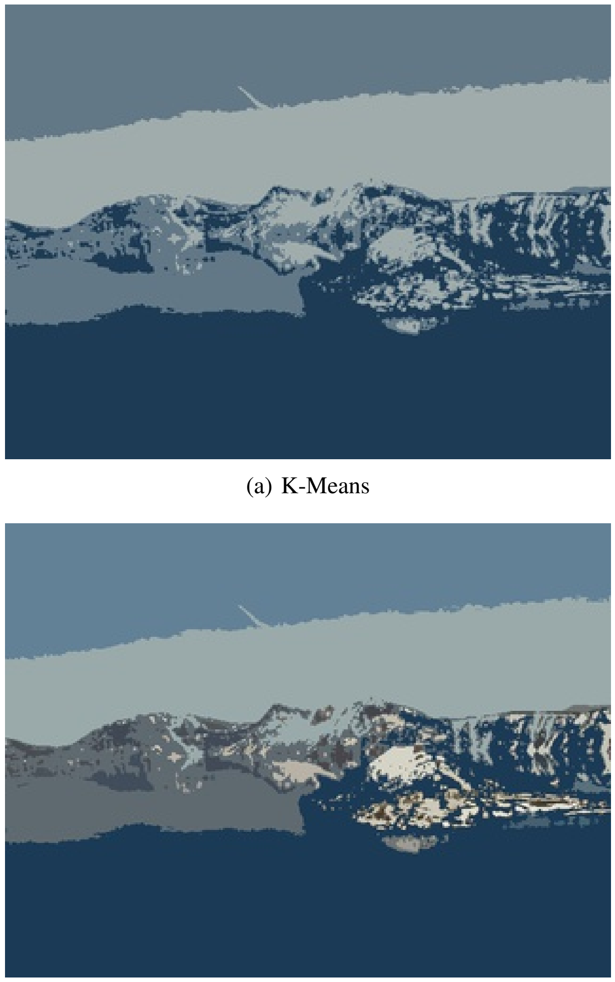
II. Image Segmentation

The images were segmented as suggested in the enunciate, using K-Means to cluster the pixels by color and then separating each cluster in connected components. Figure 1 shows examples of the two stages involved in this process: clustering by pixels color, and posterior connected components calculation. Listing 1 shows the implementation of the connected components algorithm. Figure 2 shows that, even for small values of k, a high number of segments are extracted.

```
def connected_components(img, n_clusters):
    new_img = np.zeros_like(img)
    k = 0

    for i in range(n_clusters):
        mask_b = img==i
        mask_i = mask_b.astype(np.int)

        while mask_i.sum() > 0:
            inds = np.where(mask_i == 1)
            x = inds[0][0]
            y = inds[1][0]

            dfs_mask = dfs(mask_i, x, y)

            new_img[dfs_mask==1] = k
            mask_i -= dfs_mask
            k+=1

    return new_img, k
```

Listing 1: Function for computing Precision

In order to better visualize the regions, we colorized each one of them with the mean of its pixel values. Checking in detail, we saw that a significant part of the segments were uninformative tiny regions, so we managed it by first blurring the original image, since we considered it a more natural way of reducing small regions instead of explicitly discarding them. Visually, the segmentation with blurring does not differ much from the segmentation, which suggests that, semantically, the results may not differ much, but the reduction in the number of regions is about the half, so a lot of noise is removed.


(a) K-Means

(b) Connected components

Fig. 1: Two stages of the segmentation algorithm for $K = 3$.

III. Image Description

After extracting all the segments of the image, we proceed to create a descriptor that represents the complete image. To do this, we perform two steps, first the description of the regions obtained, and second the creation of a complete image descriptor. The following subsections describe each step.

A. Region description

We decided to perform the description of the regions using two different approaches, hereby called “classic” and “LBP”.

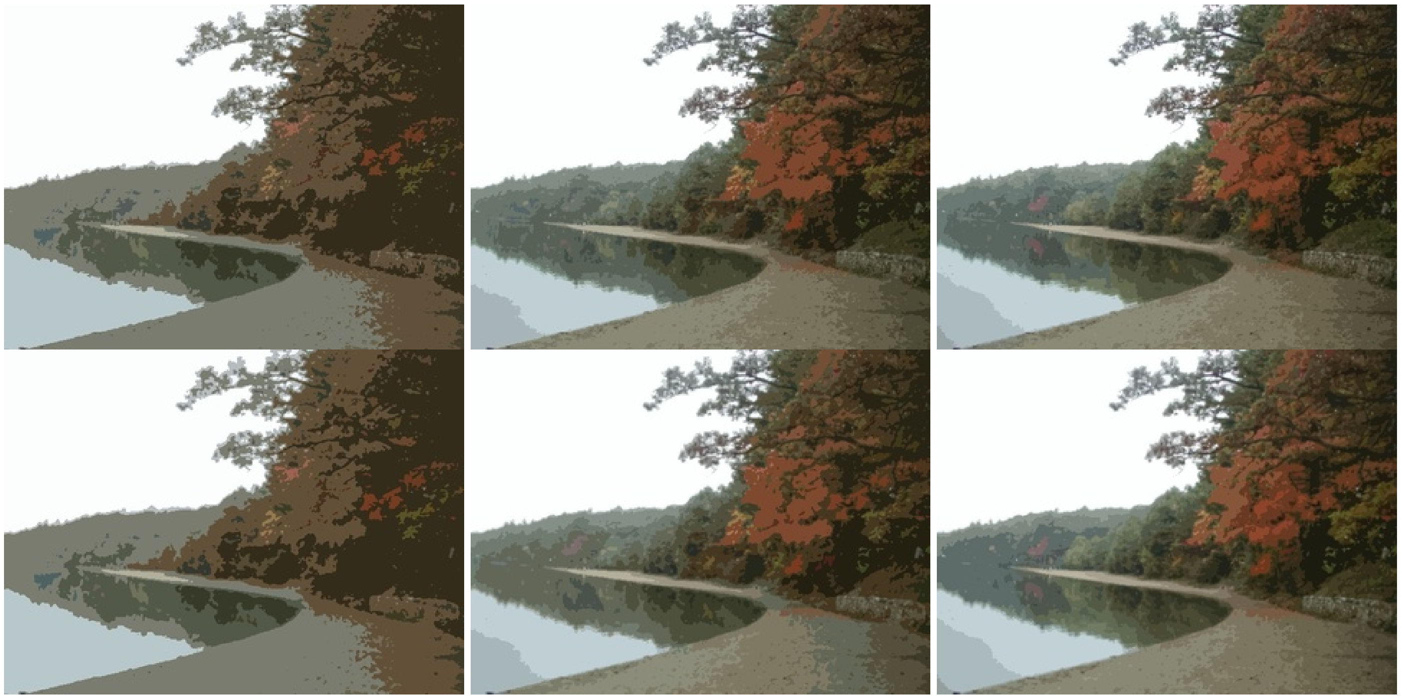

(a) $k = 5$ (b) $k = 10$ (c) $k = 15$

Fig. 2: Segmentation using different values of k. The upper row shows the segments from the original image, while the second shows the segments obtained after a blurring with a 3×3 Gaussian kernel. Although the difference is not visually evident, blurring the images caused an important decrease of the number of resulting regions after segmentation. (a) For $k = 5$, the number of segmented regions reduced from 1229 to 625. (b) For $k = 10$, the number of segmented regions reduced from 3408 to 1730. (c) For $k = 15$, the number of segmented regions reduced from 5574 to 3310.

In the "classic" description method, for each of the regions obtained from an image we create a descriptor, which is composed of spatial and texture features of the region. This descriptor is a vector concatenating the results of these different metrics. In general, since the regions can be located anywhere in the image, we decided to leave out all the features that refer to the spatial location, so we only use the spatial features that are invariant to the location.

The features used are:

- Size of the region: calculated as the number of pixels covering the region.
- Mean color: average color of the region for each channel. It is described using three numbers because the images are in RGB.
- Contrast: contrast of the region, using:

$$\sum_{i}^{Ng} \sum_{j}^{Ng} n^2 p[i,j], \; where |i-j| = n, \tag{1}$$

  where p is the co-occurrence matrix of the minimum rectangle surrounding the region. This feature is calculated for each channel.
- Correlation: correlation of the minimum rectangle that surrounds the region,

$$\frac{\sum_i^{Ng} \sum_j^{Ng} (ij) p[i,j] - \mu_x \mu_y}{\sigma_x \sigma_y}, \tag{2}$$

  where p is the co-occurrence matrix.
- Entropy: entropy of the minimum rectangle that surrounds the region,

$$-\sum_{j}^{Ng} p_i \; log_2 \; p_i. \tag{3}$$

- Bounding box: minimum rectangle that surrounds the region. We only use the height and width, due to the position of the vertex of the rectangle is not invariant to the translation.

For each one of the extracted regions from all the images, we create a representative descriptor in terms of the features previously mentioned.

The second method to describe the regions is called "LBP". This approach use the local binary pattern encoding, which consists of a binary encoding for each of the pixels depending on the neighborhood. We chose this encoding because it is very good for finding the texture patterns in an image, which helps to find better patterns of textures between regions of each image.

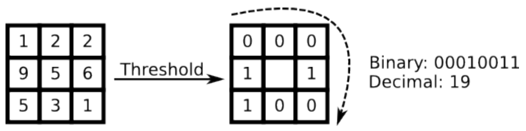


Fig. 3: Local binary patterns. The encoding is performed by means of the binarization of the neighborhood using as threshold the central value. This central value will then be updated by a new number constructed from the linear combination of the result of the neighbors.

This algorithm makes a comparison between the central pixel and its neighbors in such a way that the values that are greater than the central one take value 1 and the less they take value 0, after that a linear combination between the 0s and 1s is done in such a way that each is multiplied by 2^i, where i is the index of the neighbor, the resulting number is the new value of the central pixel. This representation can be seen visually in Fig 3. After performing the codification, we proceed to create a histogram of the region, which will be the descriptor of the region.

B. Image description

To obtain the distance between each image, it is necessary that the descriptors belong to the same vector space, so it is necessary to perform a processing on the region descriptors, because the number of regions between each of the images varies. To solve this problem we use a technique known as Bag of Visual Words (BoVW), which consists of creating a dictionary of visual words that exist in the universe of samples. The number of words is given by a constant k, which is the number of features and determines the size of the descriptors. The method consists of describing an image using a histogram of the repetitions of each of the visual words contained in the image.

To implement this algorithm we use K-Means clustering on all the descriptors of the regions of all the images. As a result of this technique, we obtained the centroids or words that define the dictionary. Then, for each of the images, a histogram is made, using as a bin each of the visual words of the dictionary and labeling each one of the regions of the image inside a bin. The result will be a k-sized descriptor, which will be the descriptor of the image.

We modeled BoVW in a very simple way, by means of hard assignment and sum pooling [1]. Instead, we could have explored the use of soft assignment and other pooling functions as well, but we will leave these investigations for future work.

IV. Image Retrieval Protocol

For the purpose of image retrieval, we validate our system according to the query images in the enunciate, which are: *beach_2, boat_5, cherry_3, pond_2, stHelens_2, sunset1_2, and sunset2_2*. We validate them by either qualitative and quantitative analysis. For the qualitative analysis, we inspected visually the results and investigate the impact of the parameters. For the quantitative analysis, we adopt *Precision* and *Average Precision* metrics.

Precision at K, or just $P@K$, is a simple and intuitive metric that measures the success rate of the query response comprising the ranked list up to position K, where K is a cut-off of interest. As the enunciate requires, we adopt K of 3. Given that we are validating an a labeled collection, we consider a retrieved image to be relevant if it belongs to the same class of the query image, and irrelevant otherwise. Listing 2 shows our implementation of the measure.

```
def precision(predictedClasses, groundTruthClass):
  return predictedClasses.count(groundTruthClass) /
    len(predictedClasses)
```

Listing 2: Function for computing Precision

Average Precision at K, or just $AP@K$, on the other side, evaluates the retrieved result prioritizing best results when they are in the top positions, so that $AP@K$ is higher in these cases. Average Precision is essentially the area under the precision-recall curve [2]. Listing 3 shows our implementation of the measure.

```
def averagePrecision(predictedClasses,
    groundTruthClass):
  score = 0.0
  hits = 0.0
  for i,p in enumerate(predictedClasses):
    if p == groundTruthClass:
      hits += 1.0
      score += hits / (i + 1.0)
  return score / len(predictedClasses)
```

Listing 3: Function for computing Precision

For every query, we retrieve the three most similar results, without considering the query itself: we guarantee the query not to be present in its result. This notion of similarity is considered as the inverse of distance, so we actually retrieve results in ascending order of distance. Given that each image is represented as an image vector, as we explained before, the ranked list is composed by vector comparisons between the query's vector to the others. As expected, the distance function plays a crucial rule. We evaluate the use of the following distance functions: Euclidean, cosine, and correlation. For these functions, we use the module *scipy.spatial.distance*[1].

V. Results

We run a series of experiments, varying the following parameters:

- The number of clusters K for segmentation, with values 5, 10, and 15.
- Whether the images was blurred before segmentation or not. For the sake of simplicity, we vary this parameter only for $K = 15$, since smaller segments are more sensitive to blurring and its impact on the segmentation is higher.
- The region descriptor was either *Classic* or *LBP*.
- The number of visual words N, with values 50, 75, and 100.

[1]https://docs.scipy.org/doc/scipy/reference/spatial.distance.html, as of October 20th, 2017

				Euclidean		Cosine		Correlation	
				Precision	Avrg Precision	Precision	Avrg Precision	Precision	Avrg Precision
K	Blur	Descriptor	N BoVW						
5	N	Classic	50	61.90 +- 41.51	60.30 +- 41.51	57.14 +- 34.34	57.14 +- 34.34	57.14 +- 34.34	57.14 +- 34.34
5	N	Classic	75	61.90 +- 41.51	60.30 +- 41.51	61.90 +- 32.99	61.90 +- 32.99	66.67 +- 25.20	64.29 +- 28.77
5	N	Classic	100	61.90 +- 37.50	59.52 +- 39.70	66.67 +- 25.20	63.49 +- 30.12	66.67 +- 25.20	64.49 +- 30.12
5	N	LBP	50	47.62 +- 30.12	31.75 +- 30.82	33.33 +- 30.86	25.40 +- 26.20	33.33 +- 30.56	25.40 +- 26.20
5	N	LBP	75	47.62 +- 30.12	31.75 +- 30.82	38.10 +- 21.30	30.16 +- 22.78	38.10 +- 21.30	30.16 +- 22.78
5	N	LBP	100	47.62 +- 30.12	31.75 +- 30.82	19.05 +- 24.28	15.08 +- 20.30	19.05 +- 24.28	15.08 +- 20.30
10	N	Classic	50	66.67 +- 25.20	65.08 +- 25.50	71.43 +- 21.30	71.43 +- 21.30	76.19 +- 15.06	76.19 +- 15.06
10	N	Classic	75	66.67 +- 35.63	65.08 +- 35.85	76.19 +- 29.35	74.60 +- 30.12	76.19 +- 29.35	74.60 +- 30.12
10	N	Classic	100	66.67 +- 25.20	62.70 +- 27.01	71.43 +- 21.30	71.43 +- 21.30	76.19 +- 15.06	74.60 +- 16.50
10	N	LBP	50	33.33 +- 25.20	30.16 +- 26.37	38.10 +- 21.30	34.13 +- 20.67	38.10 +- 21.30	34.13 +- 20.67
10	N	LBP	75	42.86 +- 15.06	31.75 +- 23.14	61.90 +- 32.99	57.14 +- 36.33	61.90 +- 32.99	57.94 +- 35.37
10	N	LBP	100	33.33 +- 25.20	30.16 +- 26.37	71.43 +- 21.30	69.84 +- 21.99	71.43 + 21.30	71.43 +- 21.30
15	N	Classic	50	**71.43 +- 21.30**	**69.05 +- 25.86**	76.19 +- 15.06	76.19 +- 15.06	76.19 +- 15.06	74.60 +- 16.50
15	N	Classic	75	66.67 +- 35.63	66.67 +- 35.63	76.19 +- 34.34	76.19 +- 34.34	**80.95 +- 34.99**	**80.95 +- 34.99**
15	N	Classic	100	71.43 +- 32.99	69.84 +- 33.45	**76.19 +- 23.33**	**76.19 +- 23.33**	76.19 +- 23.33	76.19 +- 23.33
15	N	LBP	50	33.33 +- 25.20	30.16 +- 26.37	38.10 +- 21.30	34.13 +- 20.67	38.10 +- 21.30	34.13 +- 20.67
15	N	LBP	75	42.88 +- 15.06	31.75 +- 23.14	61.90 +- 32.99	57.14 +- 36.33	61.90 +- 32.99	57.94 +- 35.37
15	N	LBP	100	33.33 +- 25.20	30.16 +- 26.37	71.43 +- 21.30	69.84 +- 21.99	71.43 +- 21.30	71.43 +- 21.30
15	Y	Classic	50	66.67 +- 30.56	63.49 +- 31.27	71.43 +- 32.99	71.43 +- 32.99	71.43 +- 32.99	71.43 +- 32.99
15	Y	Classic	75	61.90 +- 27.77	60.32 +- 27.77	61.90 +- 27.77	60.32 +- 27.77	61.90 +- 27.77	61.90 +- 27.77
15	Y	Classic	100	61.90 +- 27.77	60.32 +- 27.77	66.67 +- 30.56	66.67 +- 30.56	66.67 +- 30.86	66.67 +- 30.86
15	Y	LBP	50	33.33 +- 25.20	22.22 +- 19.47	57.14 +- 23.33	55.56 +- 23.00	57.14 +- 23.33	55.56 +- 23.00
15	Y	LBP	75	38.10 +- 27.77	25.40 +- 22.59	47.62 +- 24.28	43.65 +- 25.27	47.62 +- 24.28	43.65 +- 25.27
15	Y	LBP	100	38.10 +- 27.77	29.37 +- 26.68	61.90 +- 21.30	56.35 +- 22.31	61.90 +- 21.30	56.35 +- 22.31

TABLE I: Precision and average precision of all the experiments performed. The values in red correspond to the highest scores obtained per distance measure.

- The distances considered were Euclidean, correlation and cosine.

Table I shows the average values of the metrics introduced above for each configuration in all the queries, and Figure 4 shows the results for the best quantitative configuration. The high variability is due to the small quantity of retrieved examples, so a difference of one example means a difference of 33% in Precision. We observed that by augmenting the value of K yielded a better performance, regardless of the descriptor used and the distance, because the higher the number of segments to be clustered, the more likely to obtain good visual words.

Unlike what we supposed about the noise-removing effect of blurring, the results obtained were worse than when blurring was not applied, because it discarded discriminative information that are not necessarily noisy. Regarding the descriptors used, *Classic* outperformed *LBP* in most the cases, and for almost all the queries. Contrary to our intuition that larger dictionaries would lead to better performance, the impact of augmenting the number of visual words was not evident in almost all the cases. This could be due to an asymmetry in the contribution of the features, since larger (presumably more informative) segments contribute the same as smaller ones.

Regarding our qualitative assessment, we show in Figure 5 the configuration we considered that did the best work, with respect to appearance, which is not necessarily the same as the one with the highest scores. Although some retrieved images do not belong to the same class of the query, they actually share many visual elements with the query, suggesting that they are somehow semantically similar.

We observed that *Classic* and *LBP* descriptors were competitive with close results, alternating between the one of another as winners. In fact, there are many situations in which there is no clear advantage of one descriptor over others, which suggests that we could used them together as a composed descriptor. this can be done with concatenated vectors, for example. We have not explored this approach, but this is usually effective.

REFERENCES

[1] M. Kearns, Y. Mansour, and A. Y. Ng, "An information-theoretic analysis of hard and soft assignment methods for clustering," in *Learning in graphical models*. Springer, 1998, pp. 495–520.

[2] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

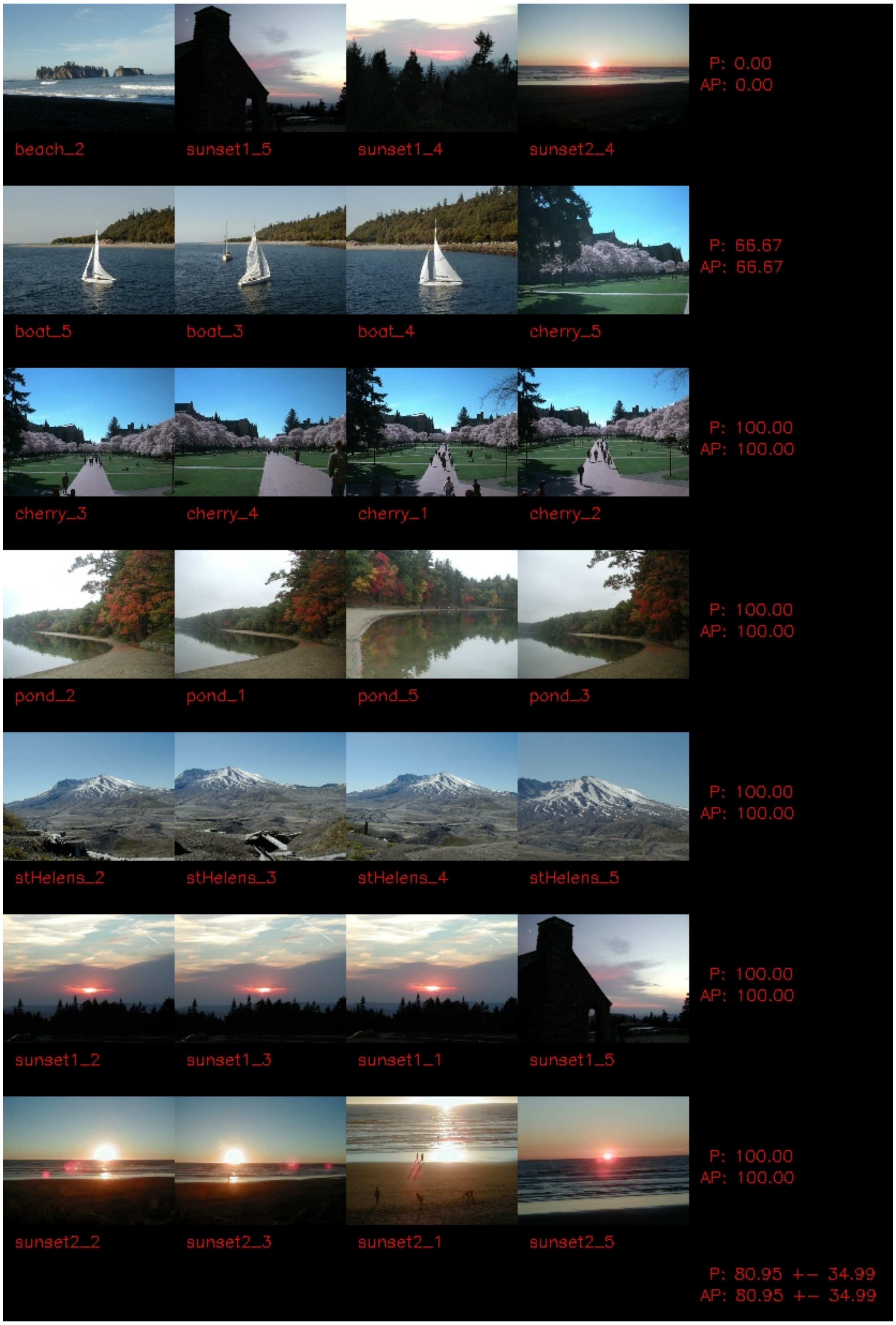


Fig. 4: Results for the configuration with the best quantitative performance (15 clusters, no blurring, *Classic* descriptor, and 75 visual words). The first column corresponds to the query and the other threes are, from left to right, the first, second and third ranked images.

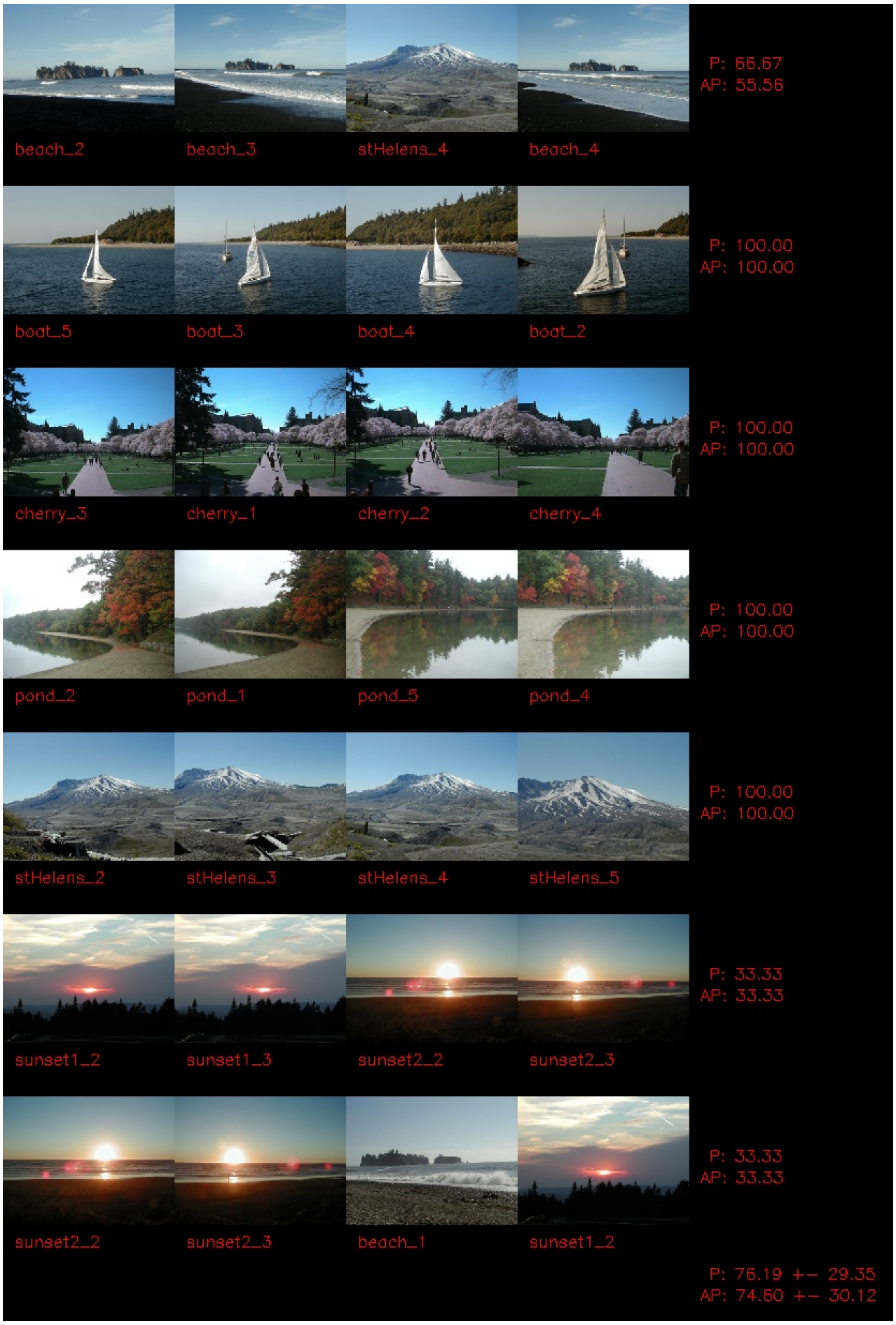


Fig. 5: Results for the configuration with a good qualitative performance (10 clusters, no blurring, *Classic* descriptor, and 75 visual words). The first column corresponds to the query and the other threes are, from left to right, the first, second and third ranked images.