ECE/CSE 576, Spring 2019 Homework 3: Content-Based Image Retrieval

Philip Pham

May 18, 2019

1 Algorithm

For each image, we performed k-means clustering with $k = 8$ for 32 iterations.[1] Connected components was then applied to segment the image into contiguous regions. Only regions that made up at least $1/256$ of the image were kept.

For each region, the following features were computed: (1) the proportion of the image occupied the region, (2) the average red, green, and blue color levels scaled to lie in range $[0, 1]$, (3) the centroid, (4) the bounding box, and (5) a normalized gray-level co-occurrence matrix (GLCM).

For the centroid and bounding box, coordinates were scaled to lie in range $[0, 1]$. The gray-levels were binned into 8 buckets each of size 32. The neighboring pixel of (r, c) was the diagonal pixel at $(r' = r + 1, c' = c + 1)$. The GLCM was initialized with entries of 0.1 which corresponds to putting a Dirichlet prior on $\text{GrayLevel}(r', c') \mid \text{GrayLevel}(r, c)$ with $\alpha = 0.1$. After counting co-occurrences, the matrix was normalized to define a proper joint probability distribution over the gray levels of a pixel and its neighbor.

1.1 Distance 1

Let $\mathbf{q}$ be the query feature vector and $\mathbf{p}$ be feature vector for another image in the database. Distance 1 was simply the squared Euclidean distance over the 5 classes of features:

$$d_1(\mathbf{q}, \mathbf{p}) = \|\mathbf{q} - \mathbf{p}\|_2^2 . \tag{1}$$

For the vectors in Equation 1, the first coordinate is the volume, the colors are the next 3 entries, the centroid are the next 2 entries in the vector, the top, right, bottom, and left of the bounding box make the next 4 entries, and the normalized GLCM are the last 64 entries for a total of $p = 74$ features per a feature vector.

1.2 Distance 2

The second distance function is more complex and uses several derived features.

$$d_2(\mathbf{q}, \mathbf{p}) = V(\mathbf{q}) \left[(V(\mathbf{q}) - V(\mathbf{p}))^2 + d_{2,\text{Color}}(\mathbf{p}, \mathbf{q}) + d_{2,\text{Position}}(\mathbf{p}, \mathbf{q}) + d_{2,\text{GLCM}}(\mathbf{p}, \mathbf{q}) \right], \tag{2}$$

where $V : \mathcal{R} \to [0, 1]$ is a function mapping a region into the percentage of the image occupied. Suppose the query vector has region vectors $\{\mathbf{q}_i\}_{i=1}^N$ and another image has region vectors $\{\mathbf{p}_j\}_{j=1}^M$. The total distance to the image given a metric d is computed as

$$D_d\left(\{\mathbf{q}_i\}_{i=1}^N, \{\mathbf{p}_j\}_{j=1}^M\right) = \frac{1}{N} \sum_{i=1}^{N} \min_j \{d(\mathbf{q}_i, \mathbf{p}_j)\}, \tag{3}$$

[1] A random seed of 2020 was used for reproducibility.

so the leading $V(\mathbf{q})$ factor in Equation 2 assigns more importance to larger regions.

Let $\text{rgb} : \mathcal{R} \to [0,1]^3$ map a region to the average color levels. Then, we define

$$d_{2,\text{Color}}(\mathbf{q},\mathbf{p}) = \left(\|\text{rgb}(\mathbf{q})\|_2 - \|\text{rgb}(\mathbf{p})\|_2\right)^2 + \|\text{rgb}(\mathbf{q}) - \text{rgb}(\mathbf{p})\|_1 + \left(1 - \frac{\text{rgb}(\mathbf{q}) \cdot \text{rgb}(\mathbf{p})}{\|\text{rgb}(\mathbf{q})\|_2 \|\text{rgb}(\mathbf{p})\|_2}\right), \tag{4}$$

where the first term penalizes differing magnitude, the second term is the l_1 distance, and the third term is the cosine distance.

Let $\text{Row} : \mathcal{R} \to [0,1]$ and $\text{Col} : \mathcal{R} \to [0,1]$ be the relative row and columns for a region. Given two region bounding boxes, we can compute the Jaccard index also know as *intersection over union*. Denote the map $J : \mathcal{R} \times \mathcal{R} \to [0,1]$. Based on the centroid and bounding box, we define

$$d_{2,\text{Position}}(\mathbf{q},\mathbf{p}) = \frac{1}{2}\left(\text{Row}(\mathbf{q}) - \text{Row}(\mathbf{p})\right)^2 + \frac{1}{2}\left(\text{Col}(\mathbf{q}) - \text{Col}(\mathbf{p})\right)^2 + \left(1 - J(\mathbf{p},\mathbf{q})\right)^2. \tag{5}$$

The first two terms penalize regions located in the different areas by their centroids. The last term uses the region bounding boxes and penalizes regions without a lot of overlap, whether because they are located in different parts of the images or their sizes differ considerably.

From the normalized 8×8 GLCM, $N_{\text{GLCM}}(\mathbf{q})$, the auxiliary features *Contrast*, *Energy*, and *Entropy* can be defined:

$$\text{Contrast}(\mathbf{q}) = \sum_{i=1}^{8}\sum_{j=1}^{8}(i-j)^2 N_{\text{GLCM}}(\mathbf{q})(i,j) \tag{6}$$

$$\text{Energy}(\mathbf{q}) = \sum_{i=1}^{8}\sum_{j=1}^{8}\left[N_{\text{GLCM}}(\mathbf{q})(i,j)\right]^2 \tag{7}$$

$$\text{Entropy}(\mathbf{q}) = -\sum_{i=1}^{8}\sum_{j=1}^{8} N_{\text{GLCM}}(\mathbf{q})(i,j)\log_2 N_{\text{GLCM}}(\mathbf{q})(i,j). \tag{8}$$

Because of differing scales and levels of importance, we defined

$$d_{2,\text{GLCM}}(\mathbf{q},\mathbf{p}) = 2\left[\text{Contrast}(\mathbf{q}) - \text{Contrast}(\mathbf{p})\right]^2 + \frac{1}{2}\left[\text{Energy}(\mathbf{q}) - \text{Energy}(\mathbf{p})\right]^2 + \frac{1}{2}\left[\text{Entropy}(\mathbf{q}) - \text{Entropy}(\mathbf{p})\right]^2. \tag{9}$$

as a weighted sum of square differences.

2 Discussion

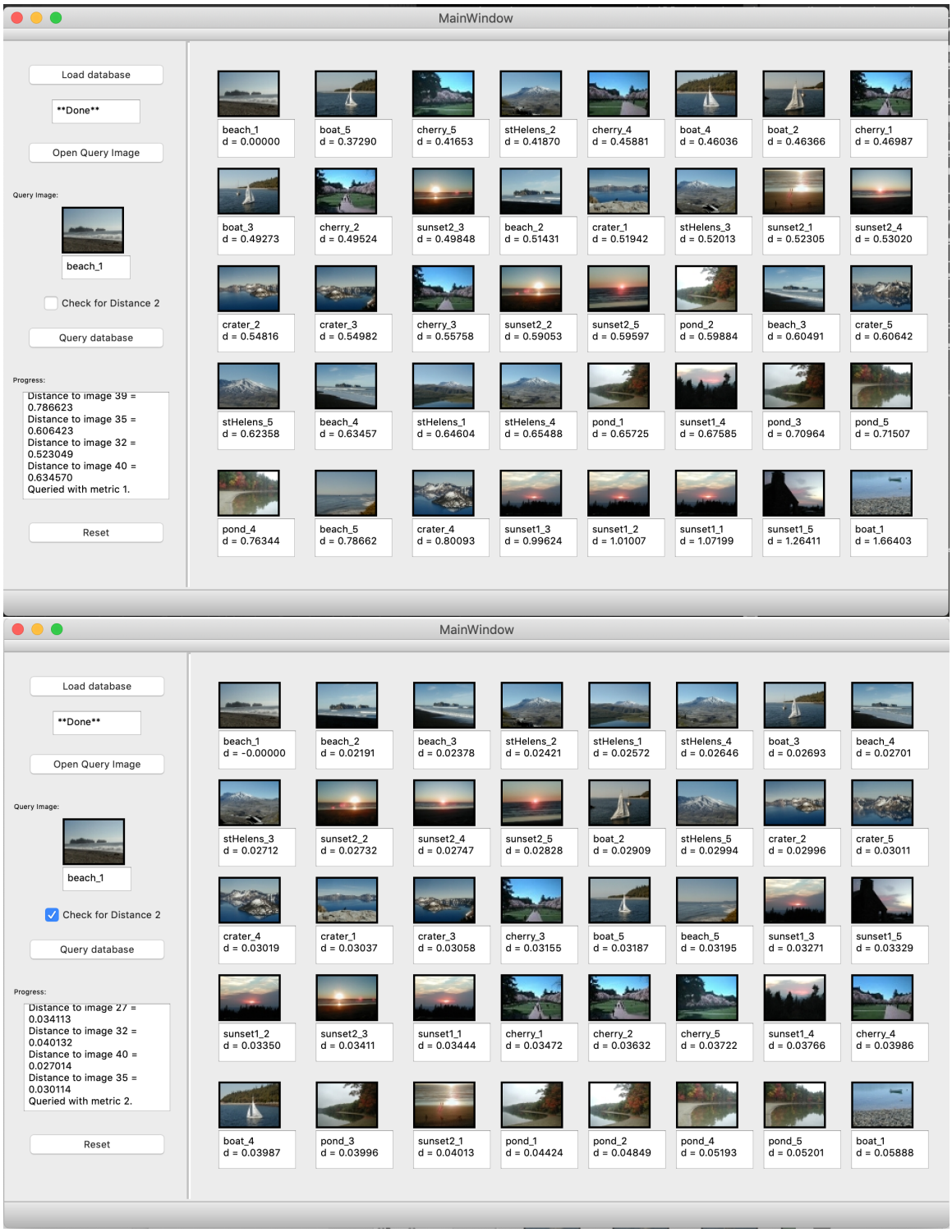
Both distances have the property that if $\mathbf{p} = \mathbf{q}$, $d(\mathbf{p},\mathbf{q}) = 0$, so if the image is in the database, the image itself will always be returned as the top result.

Color features were found to be very important which explains why a simple metric like d_1 can often be successful. In more complex images, weighting by the region size, taking into account position with the bounding box, and using contrast for texture analysis improved retrieval significantly. d_2 almost always outperformed d_1 by taking into account these features.

Results of empirical validation are shown in the next section.

3 Empirical Results

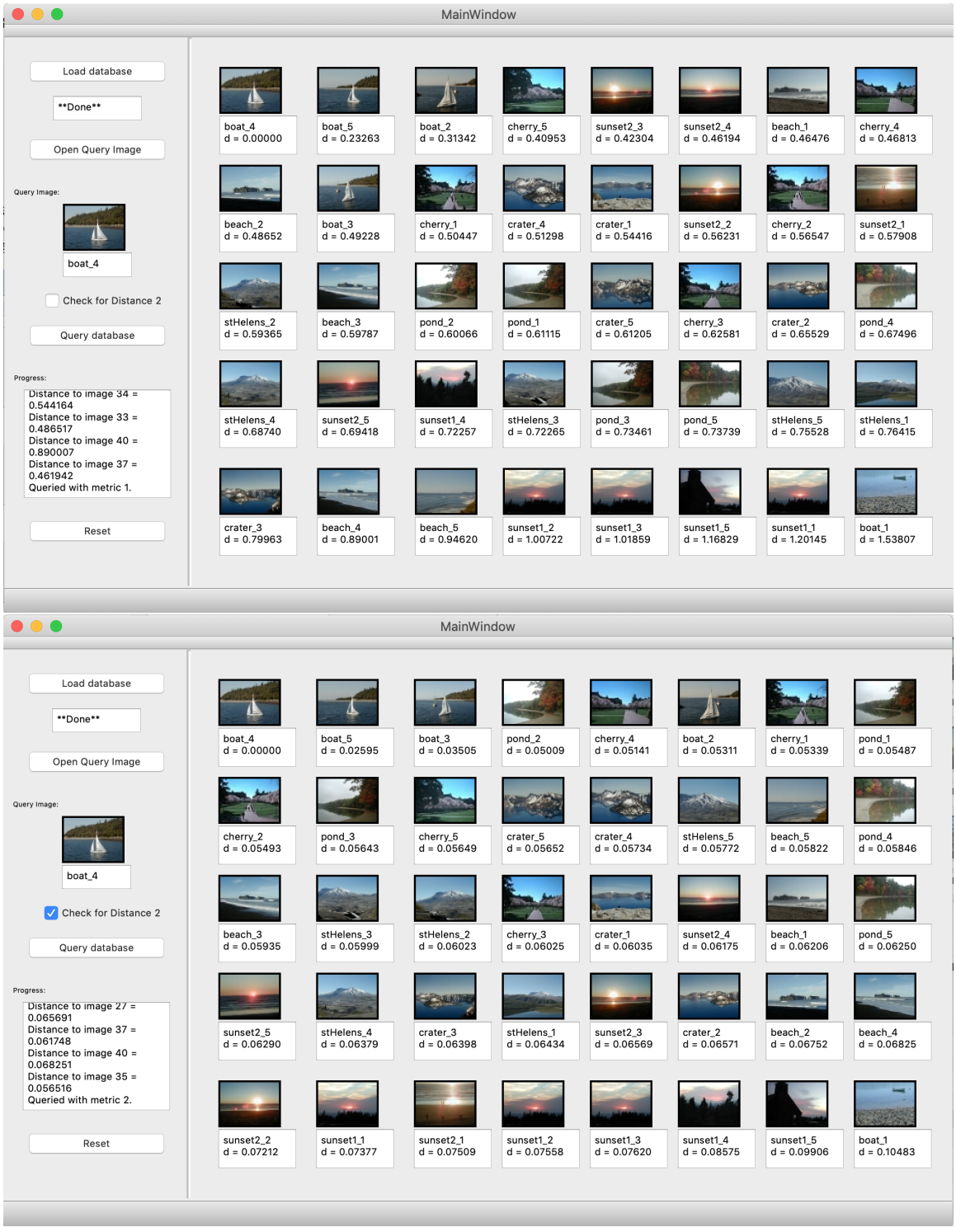
3.1 `beach`



Query results for `beach_1.jpg`.

The beach queries proved challenging. Euclidean distance completely fails, while d_2 only manages to return 2/4 beach images. Returning an additional beach image in the top row, d_2 does better.

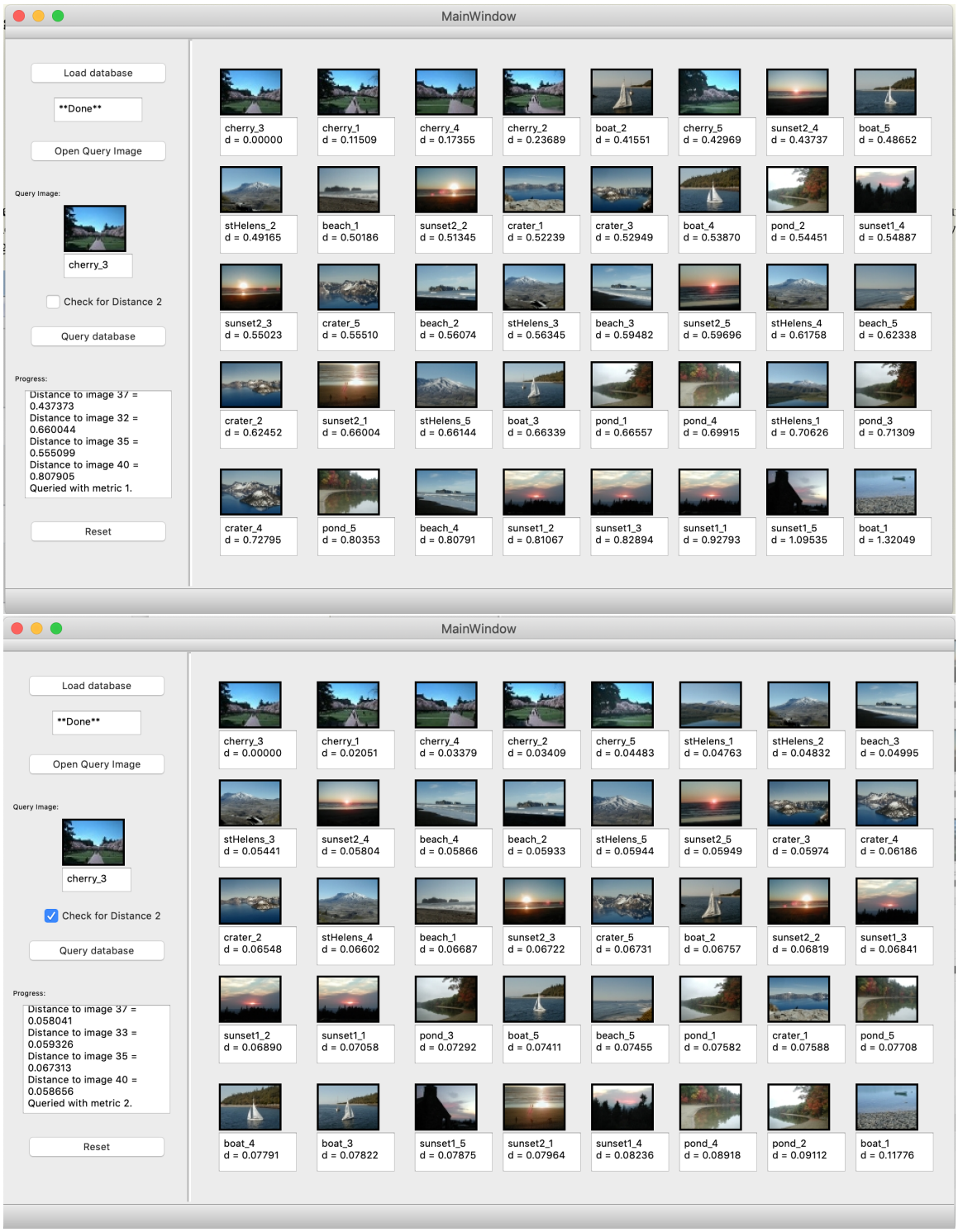
3.2 boat



Query results for `boat_4.jpg`.

d_1 and d_2 both return other boat images for their top two results. One might say that d_2 does slightly better since it has another boat result in the top row. The boat in `boat_1` is too different and not matched.

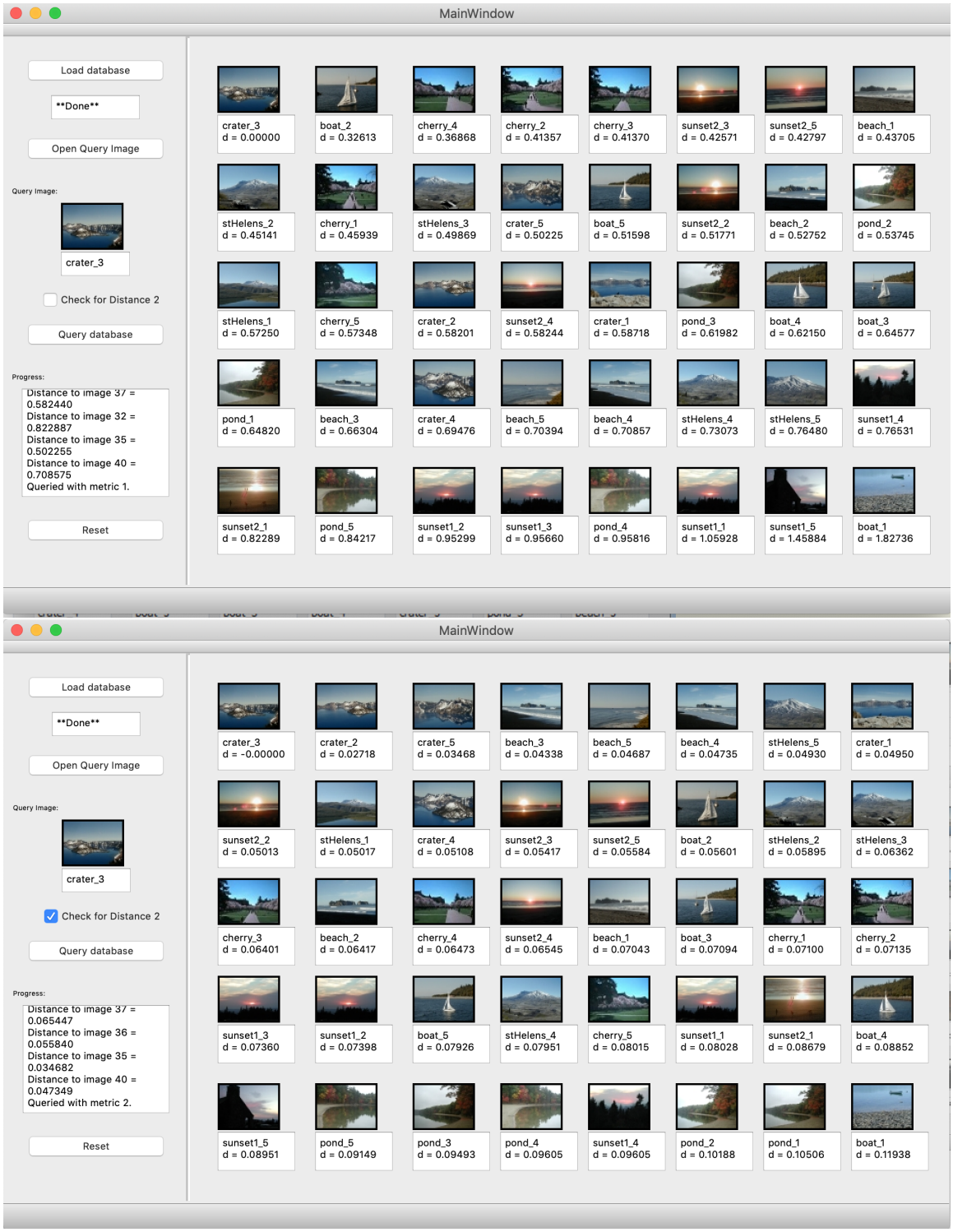
3.3 `cherry`



Query results for `cherry_3.jpg`.

These queries were among the easiest. d_1 is near-perfect barely missing `cherry_5`, while d_2 retrieves all the relevant images as its top results.

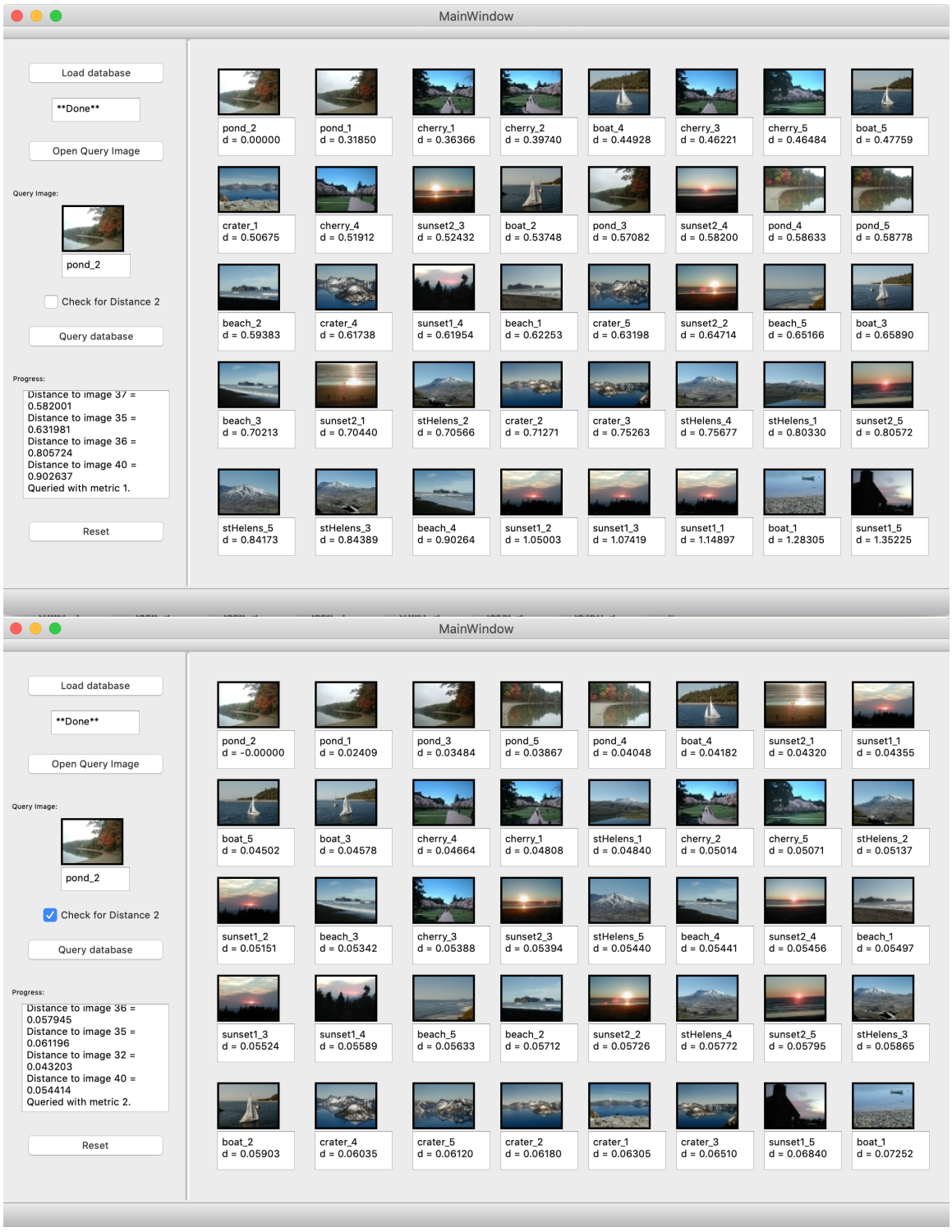
3.4 `crater`



Query results for `crater_3.jpg`.

d_1 completely fails here and doesn't return any relevant images in the top row. d_2 does respectably retrieving 2/4 crater images with an additional crater image in the top row.

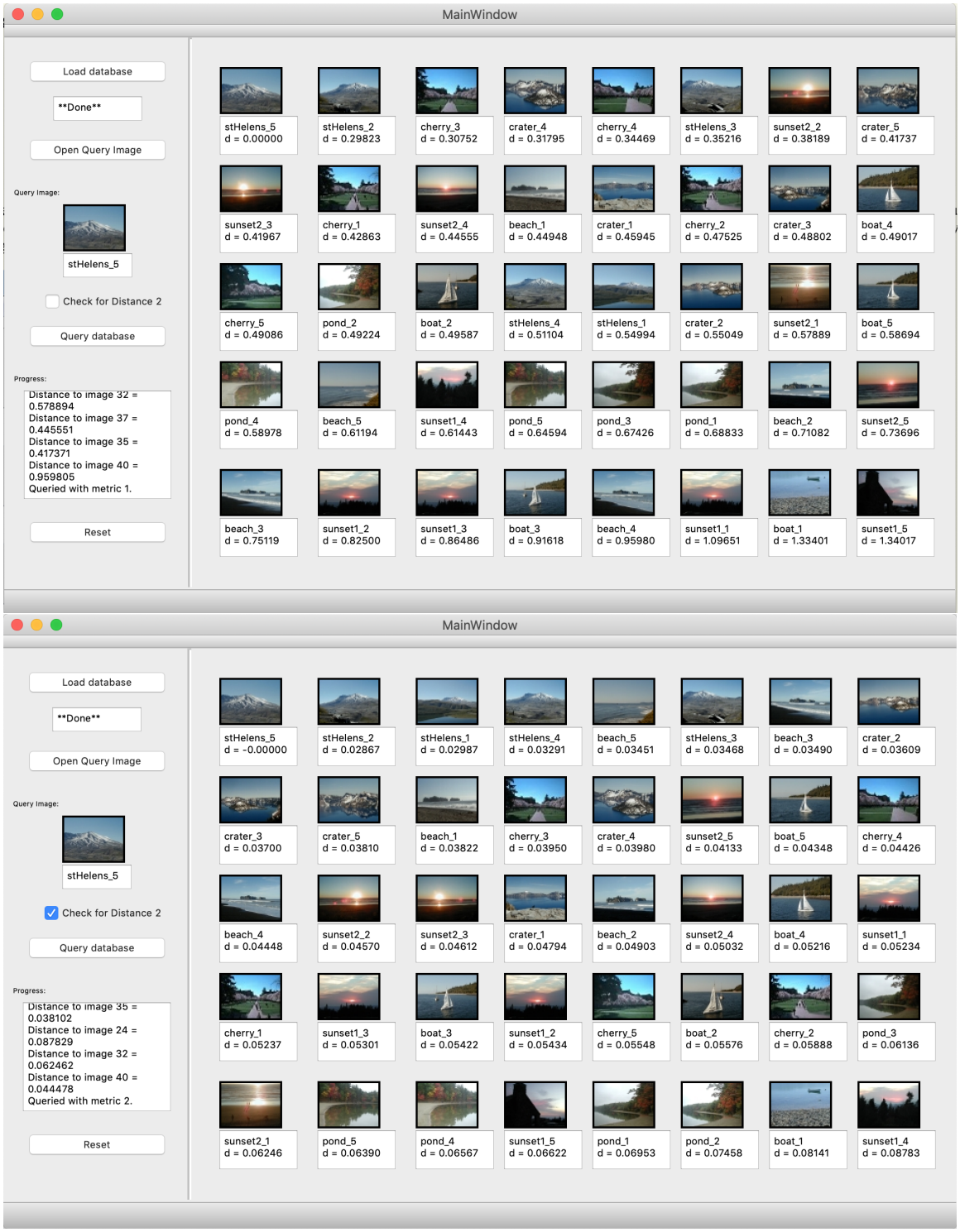
3.5 pond



Query results for `pond_2.jpg`.

d_1 mostly fails here. While its top result is another pond image, the other 3 are nowhere to be found in the top row. d_2 does perfectly: its top 4 results are the other 4 pond images.

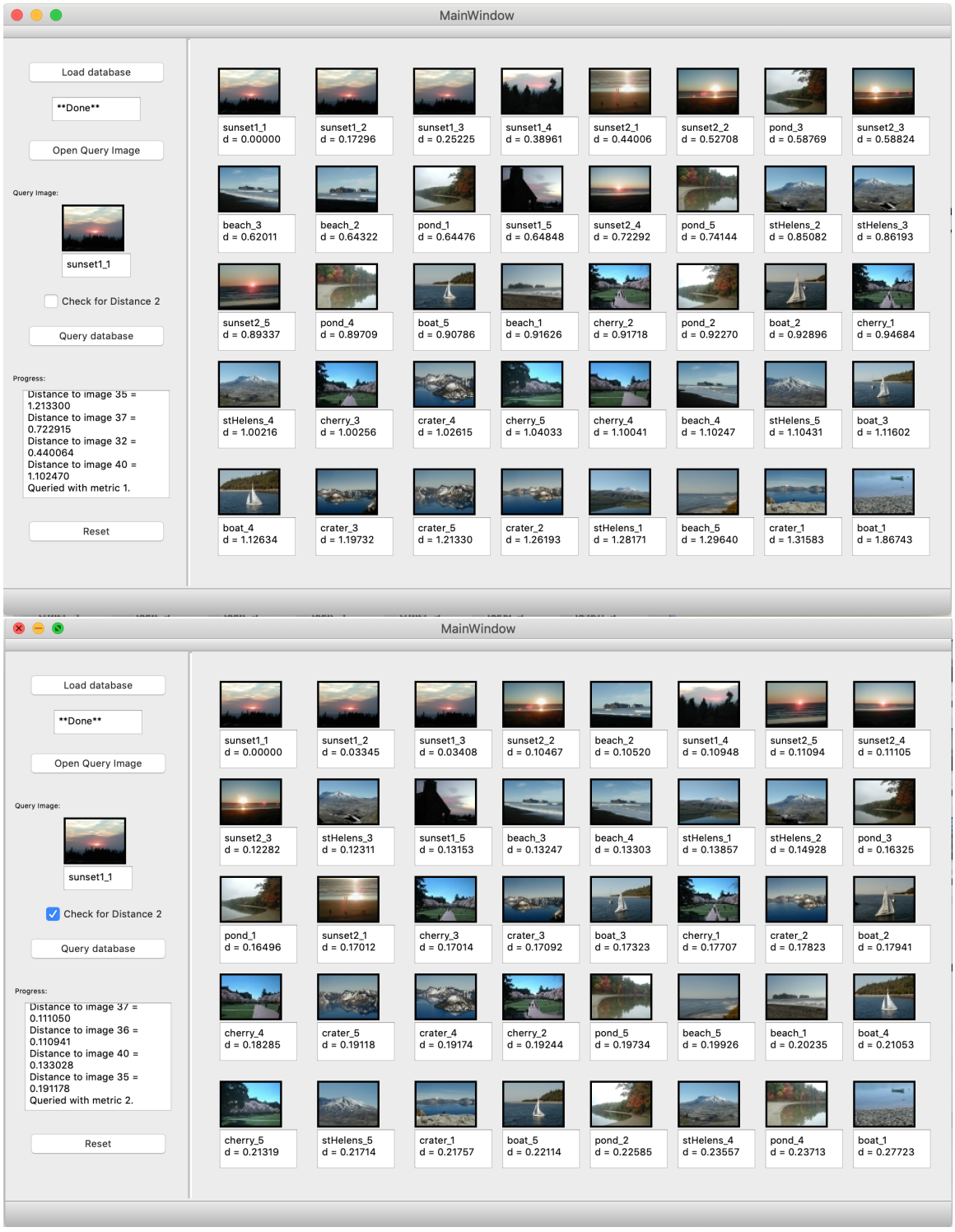
3.6 stHelens



Query results for `stHelens_5.jpg`.

d_1 does okay returning another image of Mount St. Helens as its top result with another in the top row. d_2 does very well with its top 3 results being of Mount St. Helens and narrowly missing `stHelens_3`.

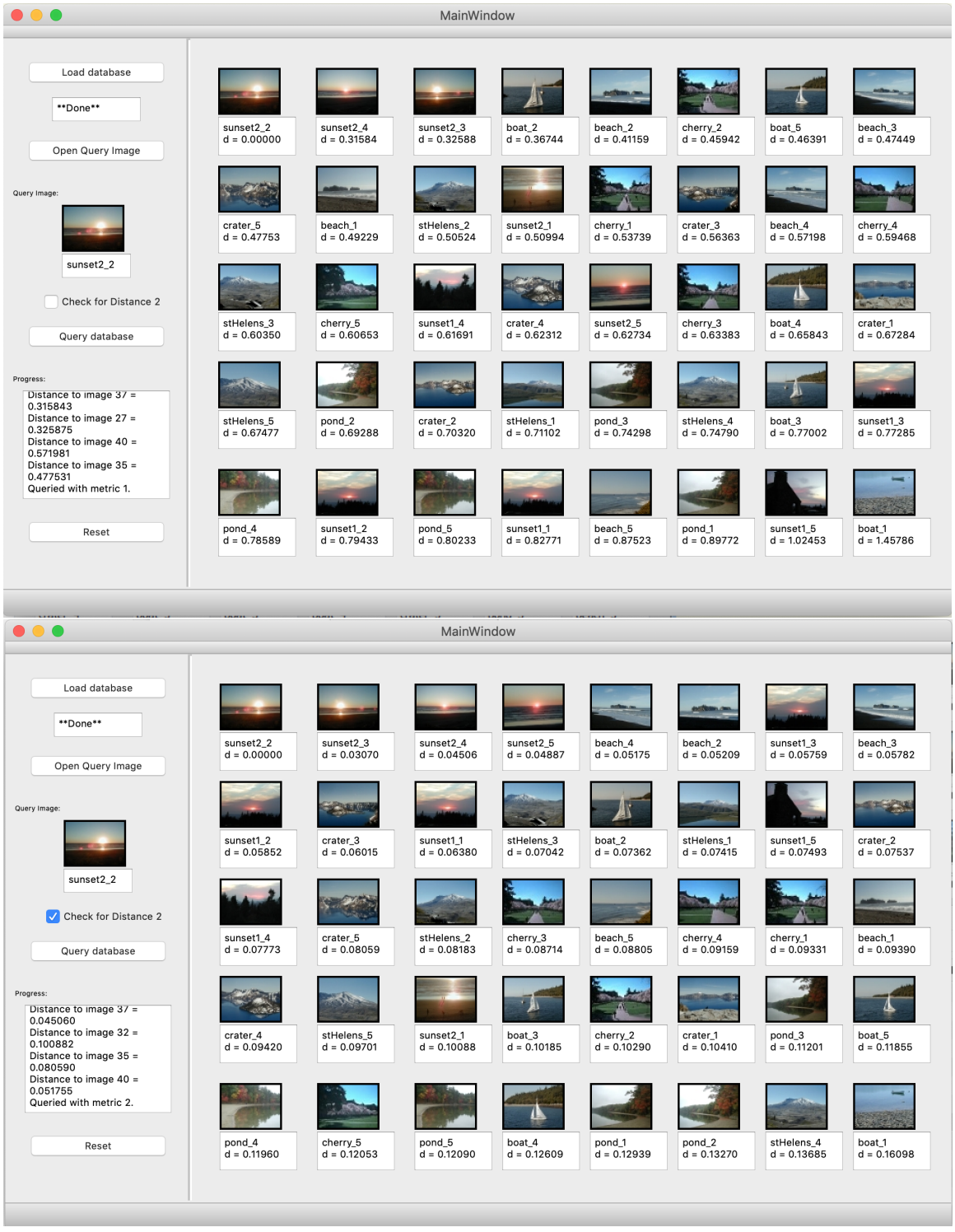
3.7 sunset1



Query results for `sunset1_1.jpg`.

Both d_1 and d_2 seem to do equally well here. The top row is all sunsets except for one image in both cases.

3.8 sunset2



Query results for `sunset2_2.jpg`.

This sunset is more challenging. d_1's top two results are sunsets, but d_2 does better: its top 3 results are sunsets with an additional result in the top row.

Appendix

All code used to generate these images can be found at `ppham27/cse576/hw3`. The embedded JPEG, PNG files, and the LaTeXcan be found in `ppham27/cse576/hw3/report`.