Appendix B

Algorithm 1: Duplicate and Near-Duplicate Image Detection

```
Input: image_list, similarity_thresh
Output: duplicate_pairs, near_duplicate_pairs, num_duplicates, num_near_duplicates

Main Detection Procedure
  1. Initialize empty lists for duplicate_pairs and near_duplicate_pairs
  2. For each pair (i, j) where 0 ≤ i < j < length(image_list):
     • If image_list[i] and image_list[j] are exactly identical:
        o Add (i, j) to duplicate_pairs
     • Else:
        o similarity_score = ORB_SIMILARITY(image_list[i], image_list[j])
        o If similarity_score > similarity_thresh and < 1.0:
           ▪ Add (i, j) to near_duplicate_pairs
  3. Count duplicates using grouping:
     • num_duplicates = COUNT_DUPLICATES(duplicate_pairs)
     • num_near_duplicates= COUNT_DUPLICATES(near_duplicate_pairs)
  4. Return all pairs and counts

Function: ORB_SIMILARITY(img1, img2)
  1. Detect ORB keypoints and descriptors for both images
  2. Match descriptors using brute-force matcher
  3. If no matches found: return 0
  4. Count strong matches (distance < 50)
  5. Return ratio: strong_matches / total_matches

Function: COUNT_DUPLICATES(pairs)
  1. Initialize a dictionary groups, creating a key for every unique element found in all pairs.
  2. For each key in groups and each pair (i, j):
     • If key matches i or j: add both to groups[key]
  3. For each group: keep all except smallest after sorting
```

4. Return count of unique extracted elements