symmetry

MDPI

Article

Deformable Object Matching Algorithm Using Fast Agglomerative Binary Search Tree Clustering

Jaehyup Jeong, Insu Won, Hunjun Yang, Bowon Lee * and Dongseok Jeong *

Department of Electronic Engineering, Inha University, Incheon 22212, Korea; jaehyup@inha.edu (J.J.); is.won@inha.ac.kr (I.W.); pinion@inha.edu (H.Y.)

* Correspondence: bowon.lee@inha.ac.kr (B.L.); dsjeong@inha.ac.kr (D.J.); Tel.: +82-32-860-7415 (B.L. & D.J.)

Academic Editor: Angel Garrido

Received: 7 November 2016; Accepted: 4 February 2017; Published: 10 February 2017

Abstract: Deformable objects have changeable shapes and they require a different method of matching algorithm compared to rigid objects. This paper proposes a fast and robust deformable object matching algorithm. First, robust feature points are selected using a statistical characteristic to obtain the feature points with the extraction method. Next, matching pairs are composed by the feature point matching of two images using the matching method. Rapid clustering is performed using the BST (Binary Search Tree) method by obtaining the geometric similarity between the matching pairs. Finally, the matching of the two images is determined after verifying the suitability of the composed cluster. An experiment with five different image sets with deformable objects confirmed the superior robustness and independence of the proposed algorithm while demonstrating up to 60 times faster matching speed compared to the conventional deformable object matching algorithms.

Keywords: content-based image retrieval; image matching; deformable object; clustering

1. Introduction

Humans can recognize and determine objects through vision. Human vision is fast and robust, and it is the most powerful perceptual function to acquire information. Vision is an ability that humans have from birth, and the human performance is far better than that of a computer. Computers may have better performance in fields that are difficult to work with human eyes, such as precision measurements. In a field of recognizing and determining objects, however, their ability is still worse than that of humans. Therefore, research to provide computers with the visual ability at the human level is currently active. Such research is called computer vision. Studies of computer vision are being performed for the recognition of face, object, gesture, from videos or images.

In image recognition, computer vision is divided into the extraction method, which belongs to low-level vision, and the matching method, which belongs to high-level vision. The typical algorithms of the extraction method include D. Lowe's SIFT (Scale-Invariant Feature Transform) [1], which is robust to size and angle change, H. Bay's SURF (Speeded Up Robust Features) [2], which is faster than SIFT, J. Matas's region-based MSER (Maximally Stable Extremal Regions) [3], and K. Mikolajczyk's Harris affine detector [4], which is robust to affine changes. The matching method is divided into a step for composing matching pairs between all the feature points of two images, and a step for performing geometric verification between the matching pairs. In particular, the geometric verification step is the final step in image recognition, and it is very important because, even if many matching pairs are composed, two images may be determined to be mutually different images if geometric verification fails. A typical algorithm for geometric verification is RANSAC [5].

In recent years, image recognition using deep learning has become popular [6]. Deep learning is different from conventional computer vision algorithms (divided into low-and high-level vision). It enables a computer to learn by itself using neural networks, without image feature extraction

and matching method, and it is leading to unparalleled levels of accuracy in image recognition. However, deep learning has not yet been used in various object matching due to the requirement for a large amount of data. With a small amount of data in a database, it is still difficult to achieve reasonably good performance for image recognition using deep learning. In addition, to detect unique objects, neural networks have to become much deeper and deeper networks require high computational power. Thus, we still need computer vision technology that uses low-level and high-level vision for image recognition.

A representative technology that uses image recognition is content-based image retrieval, which was established as the MPEG-7 standard. Recently, at MPEG-7, by constructing the CDVS (Compact Descriptor Visual Search) [7], a study was performed for content-based image retrieval, which retrieves an image fast for mobile devices. Content-based image retrieval is a technology that retrieves an image by extracting robust features even if various deformations in brightness, rotation, affine, and size, occur in the image. On the other hand, most matching algorithms perform retrieval by targeting images with rigid objects [8–10]. The object types also include deformable objects; typical examples include clothes, packs, and bags. For rigid objects, the object shapes do not change, but for deformable objects, the object shapes can change in various ways. Because of this difference, the conventional rigid object matching algorithms that are robust to images with rigid objects are not suitable for matching images with deformable objects. Therefore, developing a matching algorithm that is robust to images that contain deformable objects has become an important issue.

The three aspects of excellent matching algorithm are robustness, independence, and fast matching [11]. Robustness is a characteristic that determines that two images with the same object, even if deformation occurs in the object, must be determined to be identical. Independence is a characteristic that determines that two images with mutually different objects are different. Finally, matching is done rapidly in fast matching. If fast matching does not occur, an algorithm may not be appropriate for applications that require fast image retrieval. The most significant weakness of conventional deformable object matching algorithms is slow matching.

In this paper, these three aspects are considered to propose an optimal algorithm for the matching of two images with deformable objects. The remainder of this paper is organized as follows. Section 2 introduces the related works about image matching. In Section 3, the proposed algorithm is described by dividing it into extraction and matching methods. In Section 4, the experiment is described and its results are confirmed and analyzed from five image sets with various deformable objects. Section 5 evaluates the proposed algorithm and reports the conclusion.

2. Related Works

This section introduces well-known feature descriptors developed recently. In the past few years, a number of feature descriptors using binary features were developed. These feature descriptors which have fast feature extraction and less computational complexity are suitable for real-time image matching. This section also introduces the conventional deformable object matching algorithms. Deformable object matching algorithms use different matching methods from rigid object matching algorithms.

2.1. Recent Feature Descriptors

In recent years, binary feature descriptors such as BRIEF (Binary Robust Independent Elementary Features) [12], BRISK (Binary Robust Invariant Scalable Keypoints) [13], FREAK (Fast Retina Keypoint) [14], SYBA (Synthetic Basis) [15], and TreeBASIS [16] have been reported. BRIEF uses a binary string, which results in intensity comparisons at random pre-determined pixel locations. The descriptor similarity is evaluated using the Hamming distance. It trades robustness and independence for fast processing speed, but it is sensitive to image distortions and transformations. BRISK is a 512 bit binary descriptor using a FAST-based detector. It relies on easily configurable circular sampling patterns from which it computes a binary descriptor. It uses the distance ratio of

the two nearest neighbors to improve the accuracy of the detection of corresponding keypoint pairs. BRISK requires more computational complexity and more storage space than BRIEF. FREAK improves upon the sampling pattern and method of pair selection that BRISK uses. The features are much more concentrated near the keypoint.

SYBA uses a number of synthetic basis images to measure the similarity between a small image region surrounding a detected feature point and the randomly generated synthetic basis images. The TreeBASIS descriptor uses a binary vocabulary tree that is computed using basis dictionary images and a test set of feature region images. It provides improvements in descriptor size, computation time, matching speed, and accuracy.

2.2. The Conventional Deformable Object Matching Algorithms

The feature-based deformable object matching algorithms include transformation model-based [17], mesh-based [18], cluster-based [19] and graph-based [20] algorithms. The transformation model-based and mesh-based algorithms require high complexity and are not suitable for various deformations of objects. The graph-based algorithms have fast processing speed but relatively poor performance. The conventional deformable object matching algorithm is the ACC (Agglomerative Correspondence Clustering) algorithm [21], which uses the clustering method. This algorithm calculates the dissimilarity between clusters using the adaptive partial linkage model in the framework of hierarchical agglomerative clustering. The IACC (Improved ACC) algorithm [22] includes the feature selection method for selecting robust features. These two algorithms show good performance for deformable objects, but high complexity in the clustering process. The matching speed becomes slower with higher complexity, and it cannot be called a good matching algorithm with slow matching speed.

3. Proposed Algorithm

This section discusses the proposed algorithm. This section is divided into two subsections: the first discusses the extraction method, and the second discusses the matching method. Figure 1 shows the flow chart of the proposed algorithm, consisting of the extraction part (feature extraction and feature selection) and the matching part (the rest).

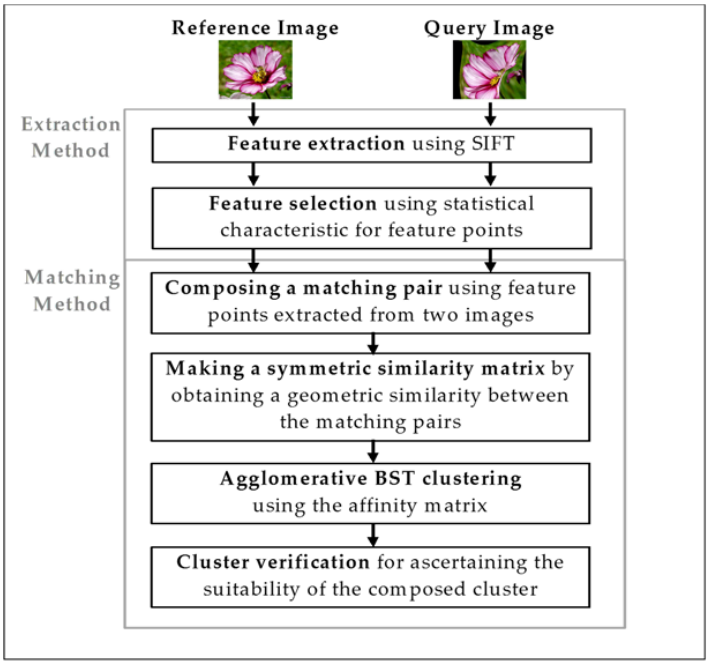


Figure 1. Flowchart of the proposed algorithm.

3.1. Extraction Method

3.1.1. Feature Extraction

There exist methods for extracting the global features and local features from images. A global feature is unsuitable for an image with deformable objects because such features are extracted from the entire image. This is because the various deformations of deformable objects cannot be defined with a single feature. On the other hand, a local feature is suitable for an image with deformable objects because the features are defined for each local region. Furthermore, a local feature is suitable for applying clustering because additional information in terms of position, scale, and orientation is stored. In this study, a typical algorithm for local features, SIFT [1], was used. The feature $F(\cdot)$ stored through SIFT is expressed as (1).

$$F(i) = \{\, p_i \,, s_i \,, o_i \,, \boldsymbol{f}_i \,\}, (1 \leq I \leq N) \tag{1}$$

where N is the number of extracted feature points, and every feature point has four components. Here, p_i is the feature point's position, s_i is the scale, o_i is the orientation, and $\boldsymbol{f}_i$ is a feature vector with 128 dimensions.

3.1.2. Feature Selection

Non-matching and higher complexity can occur if the extracted features just use matching. This is because some of the feature points could be the outliers. Therefore, it requires a process that selects the robust feature points included in the inliers. The feature selection is a process for selecting robust feature points in composing matching pairs with the extracted features. In general, when the feature points matched in two images are compared, the statistical characteristic is different between the feature points included in the outliers and those included in the inliers [23]. Therefore, the use of the inlier's statistical characteristic can distinguish the points of the inlier from the outlier. To obtain the inlier's statistical characteristics, the position (p_i), scale (s_i), orientation (o_i), and distance of the center (c_i) components are learned from various image sets [24,25]. When a large value (e_i) is produced by substituting p_i, s_i, o_i, and c_i in the learned inlier's statistical characteristic $ISC(\cdot)$, the probability of belonging to the inlier region is high. The following pseudocode shows a process for selecting N_S feature points from a total of N feature points using $ISC(\cdot)$. If N_S is bigger than N, N_S become N. We use N_S = 300. Figure 2b gives an example of using feature selection, and when compared with Figure 2a, where this is not used, some of the outlier points are removed. When the feature points of the outlier are removed because the complexity becomes lower, the features become more robust and the matching speed becomes faster.

Feature selection
$E = \{\text{ø}\}$, $i = 0$
repeat
$i = i + 1$
$e_i = ISC(p_i, s_i, o_i, c_i)$
Insert e_i into E
E, ranked in descending order
until $i = = N$
$E = \{e_1, e_2, e_3, \ldots, e_{N_S} \ldots, e_N\}$
Selecting N_S feature points from N feature points.

Figure 2. Example of the feature points in an image: (**a**) feature points using only SIFT; and (**b**) the feature points using feature selection.

3.2. Matching Method

3.2.1. Composing a Matching Pair

To compose a matching pair, the feature points extracted from two images are compared [26]. The formula used here is the Euclidean distance, as expressed in Equation (2).

$$Euclid\left(\boldsymbol{F}_{R(i)}, \boldsymbol{F}_{Q(j)}\right) = \sqrt{\sum_{k=1}^{128} \left(F_{R(i)}^{k} - F_{Q(j)}^{k}\right)^{2}} \tag{2}$$

Equation (2) is an equation for finding the Euclidean distance of $\boldsymbol{F}_{R(i)}$, which is the ith feature vector of the reference image, and $\boldsymbol{F}_{Q(j)}$, which is the jth feature vector of the query image. If $Euclid(\cdot)$ is smaller than an arbitrary threshold, the feature points $R(i)$ and $Q(j)$ are composed as a matching pair. One feature point can compose up to the maximum of k matching pairs using the *knn* method. N_M matching pairs composed in this manner undergo the overlap checking process expressed as Equation (3).

$$ovlp[i,j] = \begin{cases} 1, \text{ if } m_i \text{ and } m_j \text{ are overlapping,} \\ 0, \text{ otherwise.} \end{cases} \quad (1 \le i,\ j \le N_M) \tag{3}$$

A matching pair (m_k) is composed with two feature points matched in two images. In other words, m_k consists of the respective feature points from the reference and query images. In Equation (3), m_k represents the respective positions of two feature points. Here, $m_k = ({p^R}_k, {p^Q}_k)$, where ${p^R}_k$ is the position of the feature point extracted from the reference image, and ${p^Q}_k$ is the position of the feature point extracted from the query image. When the ith matching pair (m_i) and jth matching pair (m_j) are compared, if ${p^R}_i$ matches ${p^R}_j$, or ${p^Q}_i$ matches ${p^Q}_j$, they are determined to be overlapped, and the number one is assigned to $ovlp[i,j]$. With this equation, one or zero is assigned to every $ovlp[i,j]$, and finally, an overlap matrix of size $N_M \times N_M$ with $ovlp[i,j]$ for all i,j as its elements is generated. In Figure 3, the circles mean the feature points and lines mean the matching pairs. In addition, dotted lines are overlapped matching pairs and the solid-lines are non-overlapped matching pairs. The generated overlap matrix is used in the clustering process.

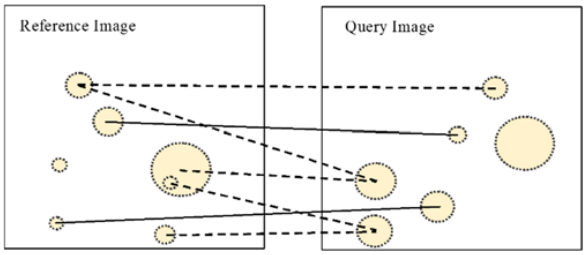


Figure 3. Example of matching pairs that overlap or not.

3.2.2. Making a Symmetric Similarity Matrix

With a deformable object, various deformations may occur because its shape can change. Therefore, it is difficult to evaluate image matching with deformable objects using conventional geometric verification. From the matching pairs composed of the typical conventional geometric verification RANSAC [5], a transform matrix is generated and inliers and outliers are distinguished. On the other hand, a deformable object cannot be defined with a single transform matrix.

Figure 4a presents two images with rigid objects, one of which has one transform matrix (T_1). The reference image's rigid object is transformed geometrically to T_1 in the query image. On the other hand, Figure 4b shows two images with deformable objects, and has many transform matrices (T_2, T_3, and T_4). In this case, a deformable object of the reference image is transformed geometrically to T_2, T_3, and T_4, in the query image. Therefore, because a deformable object cannot be defined with one transform matrix, a new method is required for the approach by generating many transform matrices in a small region. One method used here is to make a symmetric similarity matrix. The symmetric similarity matrix consists of the similarity between transform matrices composed in a point unit. In other words, a symmetric similarity matrix is composed of geometric similarity between all matching pairs.

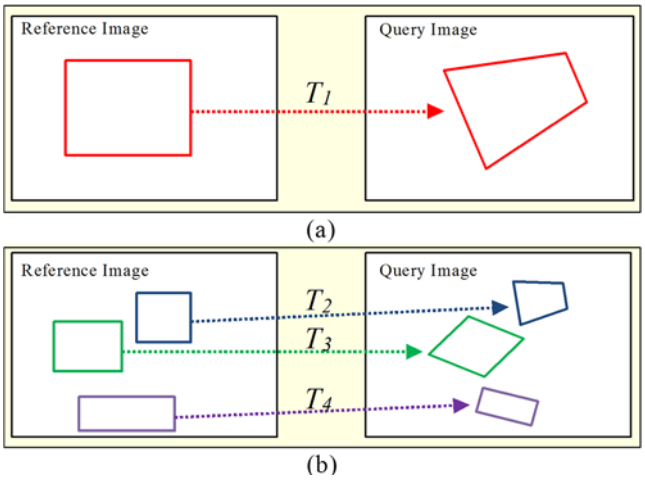
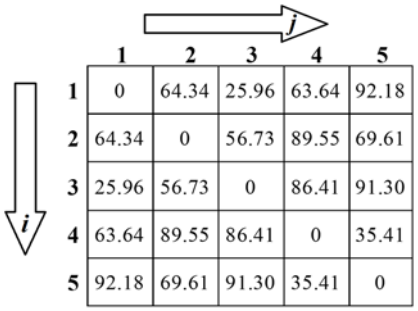


Figure 4. Comparison example of a transform matrix (T_i): (**a**) rigid object in the images; and (**b**) deformable object in the images.

To find the geometric similarity between a matching pair, first, a transform matrix is obtained between a matching pair. The transform matrix used here is a homography matrix [27]. Because a homography matrix uses the projective transform method among various transform matrices, it is suitable for obtaining geometric similarity. To compose a homography matrix, the position (p_i), scale (s_i), and orientation (o_i) of a feature point are used, and the matrix is composed using the WGC (Weak Geometric Consistency) [28] method. Using the homography matrix (H_k) composed this way, the geometric similarity (d_{gs}) between a matching pair is found using the Pairwise-WGC [29] method, as expressed in (4).

$$d_{gs}(m_i, m_j) = \frac{1}{2}\left(\left|{p^Q}_j - H_i {p^R}_j\right| + \left|{p^Q}_i - H_j {p^R}_i\right|\right), (1 \le i,\ j \le N_M) \tag{4}$$

The two matching pairs to be compared are given as $m_i = ({p^R}_i, {p^Q}_i, H_i)$ and $m_j = ({p^R}_j, {p^Q}_j, H_j)$. $|\cdot|$ denotes the Euclidean distance, and $d_{gs}(m_i, m_j)$ is small if H_i and H_j are similar. If geometric similarity is obtained between every matching pair, a symmetric similarity matrix of size $N_M \times N_M$ with $d_{gs}(m_i, m_j)$ as the element is composed, as shown in Figure 5. The symmetric similarity matrix has zero diagonal elements.



	1	2	3	4	5
1	0	64.34	25.96	63.64	92.18
2	64.34	0	56.73	89.55	69.61
3	25.96	56.73	0	86.41	91.30
4	63.64	89.55	86.41	0	35.41
5	92.18	69.61	91.30	35.41	0

Figure 5. Example of a symmetric similarity matrix (N_M = 5).

$$d_{gs}(m_i, m_j) = sim(i, j) \tag{5}$$

As written in Equation (5), each element of a symmetric similarity matrix represents geometric similarity (d_{gs}) between a matching pair m_i and m_j, and means the similarity (sim) between i and j. Here, the i and j indices become the minimum units for clustering.

Simply composing a symmetric similarity matrix does not mean a new geometric verification. The new geometric verification intended here refers to everything, from using the composed symmetric similarity matrix, to finally performing the cluster verification after undergoing the clustering process.

3.2.3. Agglomerative BST (Binary Search Tree) Clustering

For clustering, agglomerating clusters by identifying the similarities between the cluster hierarchically is common. The methods for identifying the similarity between clusters include AGNES using the single-link, complete-link, and average-link methods [30]. In the ACC and IACC algorithm [21,22], clustering is performed adaptively using the adaptive partial link method. These clustering methods, however, have a large limitation in that the speed decreases with increasing number of clusters. In general, when the number of initial clusters is n, the hierarchical clustering method has a complexity of $O(n^3)$ because the similarity between clusters needs to be calculated and updated. Here, updating means obtaining a new similarity between an agglomerated cluster and the remaining clusters. The complexity of the similarity calculation between clusters can be reduced using the symmetric similarity matrix obtained earlier, but an additional calculation is essential in the case of an update. In this paper, an algorithm is proposed to reduce the complexity by simplifying the conventional agglomerative hierarchical clustering. The update process that comprises a large proportion of the complexity is omitted, and clustering is performed by constructing a BST (Binary Search Tree) [31] with the basic clusters obtained from symmetric similarity matrix.

The pseudocode presented earlier shows the BST clustering process in detail. In the initialization part, N_{tree} is the number of binary trees (BT_t) generated, and BT_t represents the tth binary tree. The BST clustering process that appears hereon is performed the maximum of N_{bc} times. N_{bc} is the number of $sim(i, j)$ in the upper triangular part, excluding the diagonal elements in the symmetric similarity matrix, and $N_{bc} = \frac{N_M \times N_M - N_M}{2}$. When the BST clustering process is examined, first, i and j with minimum similarity are found in the symmetric similarity matrix (because the symmetric similarity matrix is a symmetrical matrix, they are found only when $i > j$). Here, BST clustering is terminated if the similarity is larger than the given threshold δ_s (similarity threshold). Next, an element of the overlap matrix with i and j as the index is confirmed. If the value for $ovlp[i, j]$ is one, clustering is not formed because the feature point with an overlap between positions cannot be considered as a robust feature.

Agglomerative BST Clustering
N_{tree} = 0, k = 0, BT_t = {ø}, $sumS$ = 0 // Initialization
/* BST clustering */
repeat
$k = k + 1$
// Find i,j $\{i, j\} = \underset{i>j}{\text{argmin}}$ (symmetric similarity matrix)
if $sim(i,j) > \delta_s$ then {**break**}
// overlap check
if $ovlp[i,j]$ then {$sim(i,j) = \infty$, **continue**}
// Using BST, Searching & Inserting
chk = 0, t = 0
repeat
if $\{i,j\} \in BT_t$ then {chk = 1, **break**}
else if $i \in BT_t$ then {Insert j into BT_t , chk = 1, **break**}
else if $j \in BT_t$ then {Insert i into BT_t , chk = 1, **break**}
else {$t = t + 1$}
until $t == N_{tree}$
// make new BT_t
if $chk == 0$ and $sim(i,j) < thres(\delta_s, sumS)$ then {
Make BT_t and Insert i,j into BT_t
$N_{tree} = N_{tree} + 1$
$sumS$ += $sim(i,j)$ }
$sim(i,j) = \infty$
until $k == N_{bc}$
if any one of the nodes in BT_t ($0 \leq t \leq N_{tree}$) is the same, merges them.
The rest of BT_t is cluster $\boldsymbol{C}_t$ ($0 \leq t \leq N_{cluster}$)

In the next part, searching and inserting i and j is performed using BST. This process is performed the maximum of N_{tree} times, and if a node is searched at least once in BT_t, it is terminated. In total, there are three cases of nodes searched from BT_t. The first is the case where both i and j are searched. Here, because all pertinent nodes exist, the process is terminated without insertion. Next is a case where only i is searched. Here, j is inserted as a new leaf node in BT_t, and the process is terminated. Finally, in the case where only j is searched, i is inserted as a new leaf node, and the process is terminated. Figure 6 gives an example of the searching and inserting process of BT_t. For example, when the i = 8 and j = 35, Figure 6a shows that the node 8 of BT_0 is searched. This is the case where i is searched. As shown in Figure 6b, j = 35 is inserted as a new leaf node in BT_0 because j is not searched in BT_0.

$$thres(\delta_s, sumS) = \frac{\delta_s}{sumS/(N_{tree}+1)} \tag{6}$$

A new BT_t is generated when t = 0 or searching is not done. To generate a new BT_t, an additional threshold is required. The root node (first node) is important for generating binary trees. If the root node is incorrect, binary tree generated from the root node can generate large errors. The additional threshold makes the root node more robust. As written in Equation (6), it is an adaptive threshold. Because $sim(i,j)$ increases as BT_t is generated, threshold must also increase. The adaptive threshold is the value that divides similarity threshold (δ_s) by the mean of the sum of root node's similarities. In the BT_t generated here, i and j are inserted as new nodes. Next, it finds new i and j with the minimum similarity value again by providing $sim(i,j) = \infty$ and clustering is repeated the maximum of N_{bc} times. Finally, it checks whether to merge between the generated binary trees. If any one of the nodes in the generated binary trees is the same, they are merged. To merge or not, all the rest of BT_t generated this way become cluster $\boldsymbol{C}_t$ with the basic clusters. For example, in BT_5 of Figure 6, because

all nodes form a basic cluster, C_5 = {7,6,60,42,28,44}. The clusters C_t generated this way finally undergo cluster verification.

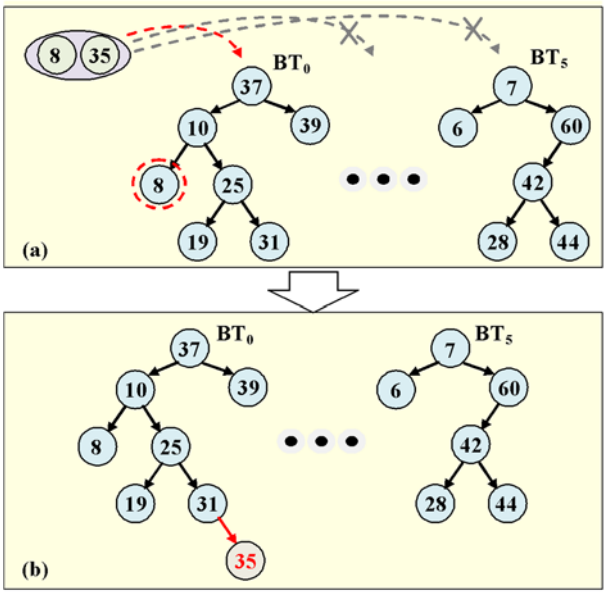


Figure 6. Example of binary search tree (t = 5). The circles in blue indicate the nodes in BT_t and the oval in purple indicate two candidate node {i = 8, j = 35}. (**a**) Node 8 is searched in BT_0 (red dotted arrow and circle); (**b**) Node 35 is inserted as a new leaf node in BT_0 (red solid arrow and red number in the circle).

3.2.4. Cluster Verification

Finally, in the matching method, the cluster verification step determines the suitability of the clusters C_t obtained as described earlier. This step is required because even if a cluster is agglomerated by the geometric similarity between the basic clusters, there is still the possibility of error. In particular, this must be considered when the cluster area is too small when the possibility of error is high. Figure 7 gives examples of mismatching results without using cluster verification, where the cluster area is too small compared to the entire image area.

Figure 7. Examples of mismatching results without using cluster verification.

Cluster Verification
cluster C_t, $t = 0$
$\text{area}_{\text{img1}}$ = entire reference image(=img1) area
$\text{area}_{\text{img2}}$ = entire query image(=img2) area
repeat
$\{\mathbf{cv}_{\text{img1}}, \mathbf{cv}_{\text{img2}}\}$ = find each convex-hull in C_t
$\text{ratio}_{\text{img1}}$ = (calculate area of $\mathbf{cv}_{\text{img1}}$)/$\text{area}_{\text{img1}}$
$\text{ratio}_{\text{img2}}$ = (calculate area of $\mathbf{cv}_{\text{img2}}$)/$\text{area}_{\text{img2}}$

$q_{min} = \min(\text{ratio}_{\text{img1}}, \text{ratio}_{\text{img2}})$
$q_{\text{ratio}} = q_{min}/\max(\text{ratio}_{\text{img1}}, \text{ratio}_{\text{img2}})$
q_{size} = the number of elements in C_t

if $q_{min} > \tau_{min}$ and $q_{\text{ratio}} > \tau_{ratio}$
and $q_{size} > \tau_{size}$ then {C_t is TRUE}
$t = t + 1$
until $t == N_{cluster}$

The previous pseudocode shows the proposed cluster verification step. Cluster verification obtains the determination criteria based on the ratio between the entire image area and the cluster area. The cluster area is calculated by obtaining a convex hull from the positions of the feature points. Here, the feature points can be obtained from the indices that correspond to each element of cluster C_t. Using the ratio that can be obtained from both the reference and query images, the minimum value q_{min} and ratio q_{ratio} of the minimum and maximum values are obtained. As another criterion, q_{size}, the number of elements of C_t, is obtained. These three determination criteria and respective thresholds, τ_{min}, τ_{ratio}, and τ_{size}, are compared, and when they are all larger than the respective thresholds, the pertinent cluster C_t is determined to be suitable. If at least one is determined to be suitable from the clusters, C_t, two images are finally determined to be matching.

4. Experiment

4.1. Experiment Conditions

To evaluate the matching performance, an experiment was performed with five types of image sets. Among these, two types were image sets that contain actual deformable objects, and the other three types were image sets where the images become artificially deformable using TPS (Thin-Plate-Spline). As shown in Figure 8, the image sets that contain actual deformable objects were composed of clothes and snack packs, which are commonly encountered in real life. For the image sets that uses TPS, Stanford University's SMVS standard images [32] and some of the ImageNet's Natural images (flowers, trees, leaves,) [33] and Oxford University's buildings images [34] were used. In the image set, the reference images were constructed with those images where a feature that could represent an object appears at the front. In the case of query images, they were constructed with the images of clothes where a person wears the clothes in various poses; images of snack packs, where various deformations are applied due to the contents in the snack packs; and SMVS and IN-N (ImageNet's Natural), and Oxbuild (Oxford building images), where warping is applied based on several arbitrary points using TPS. Table 1 lists the composition of the five types of image sets. The annotations consist of images, matching pairs of images, and non-matching pairs of images.

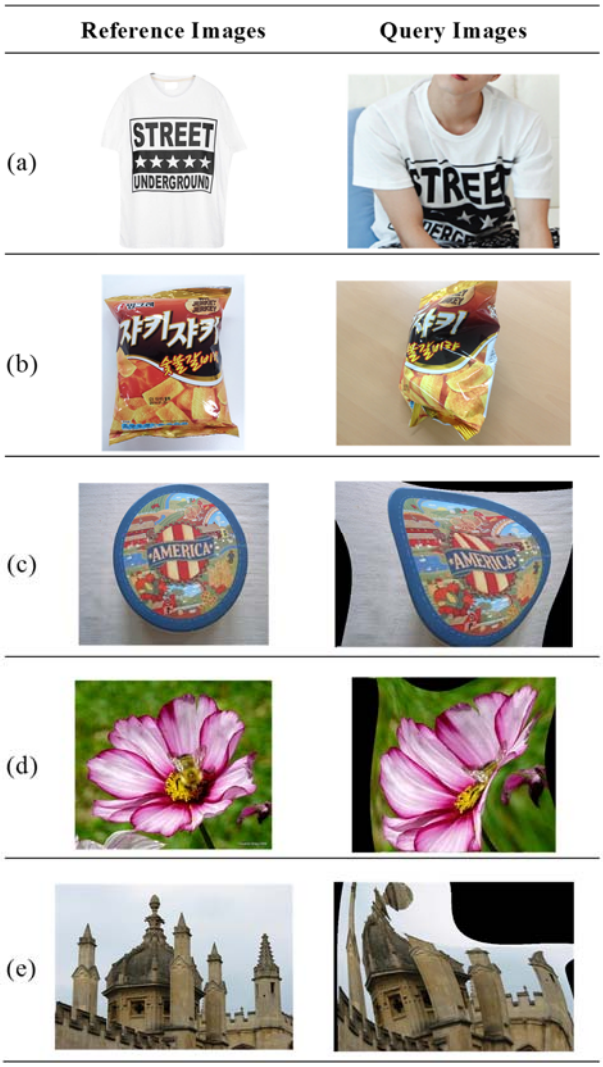


Figure 8. Examples of reference and query (deformable) images: (**a**) clothes; (**b**) snack packs; (**c**) SMVS (using TPS); (**d**) IN-Natural (using TPS); and (**e**) Oxbulid (using TPS).

Table 1. Configuration of image set.

Image Set	Annotations
Clothes	1250 images 996 matching pairs of images 4233 non-matching pairs of images
Snack packs	400 images 300 matching pairs of images 3000 non-matching pairs of images
SMVS (using TPS)	20,400 images 6576 matching pairs of images 7805 non-matching pairs of images
IN-N (using TPS)	1246 images 623 matching pairs of images 5598 non-matching pairs of images
Oxbuild (using TPS)	5063 images 5063 matching pairs of images 20,252 non-matching pairs of images

To measure the proposed algorithm performance, TPR (True Positive Rate) in Equation (7) and FPR (False Positive Rate) in Equation (8) were used. TPR is an equation for finding the robustness among the algorithm characteristics; a larger value indicates better performance. On the other hand, FPR is an equation for finding the independence among the algorithm characteristics; a smaller value indicates better performance. TPR was obtained from the matching pairs of images in Table 1, and FPR is obtained from the non-matching pairs of images in Table 1. The accuracy defined in Equation (9) represents the relationship between TPR and FPR for an objective comparison. Finally, the matching time was measured to determine the fast matching speed.

The proposed algorithm use SIFT [1] for feature extraction like the common comparison algorithms such as ACC [21], IACC [22], and RANSAC [5]. By doing this, we can compare the performance of matching method under the same conditions. In addition, SIFT showed better performance compared with the other feature descriptors such as SURF and BRISK in our experiment which is consistent with other findings [35,36] for images with various deformations. Although SIFT has slower speed for extracting features, it was determined to be an appropriate choice for the feature descriptor.

Here, the experiment was performed by applying all the major parameters required for feature extraction in SIFT. The thresholds for cluster verification were fixed as τ_{min}= 0.001, τ_{ratio}= 0.5, τ_{size}= 3.

$$TPR = \frac{TP}{TP + FN} = \frac{TP}{P} \tag{7}$$

$$FPR = \frac{FP}{FP + TN} = \frac{FP}{N} \tag{8}$$

$$Accuracy = \frac{TP + TN}{P + N} \tag{9}$$

For performance test, we used an Intel Core i5-2500 (quad core) CPU with the clock speed of 3.3 GHz and 8 GB RAM running the Windows 7(64-bit). In addition, all algorithms are implemented in the C ++ environment.

4.2. Experiment Results

Table 2 presents the average computational time and memory storage required to build and use binary trees. Compared with non-binary tree case, when δ_s increases, the algorithm runs faster; when δ_s is above 30, it is faster than non-binary tree case. Since average memory storage required to build binary trees occupies a small part of the whole memory, it is determined to be better to use binary trees.

Table 2. Requirements of the computational time and memory storage about binary tress.

δ_s	Non-Binary Tree	Use of Binary Trees	
	Average Time (ms)	Average Time (ms)	Average Memory (MB)
1	0.004	0.005	0.257
10	0.236	0.266	3.876
20	0.753	0.776	5.935
30	1.501	1.439	7.472
40	2.548	2.366	8.747
50	3.784	3.366	9.784

Figure 9 presents the top three values of accuracy (A1, A2, A3) for each algorithm using Equation (9). These are the results of experimenting with the image set of clothes, snack packs, SMVS (using TPS), IN-N (using TPS) and Oxbuild (using TPS). In the case of RANSAC, the accuracies were very low because it is not an algorithm suitable for images with deformable objects. The other algorithms showed better performance with the proposed algorithm showing the best performance.

Figure 10 presents the recall vs. precision curve using similarity threshold (δ_s) in each image set. The proposed algorithm outperformed the other algorithms, especially for high recall values.

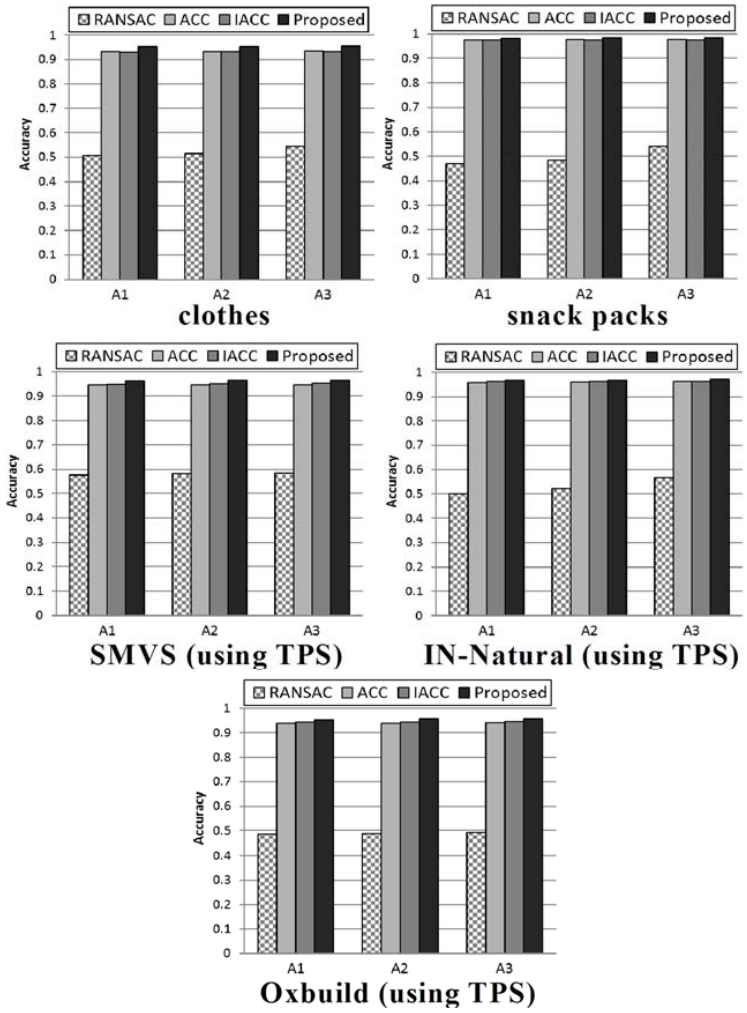


Figure 9. Accuracy of the proposed and other algorithms.

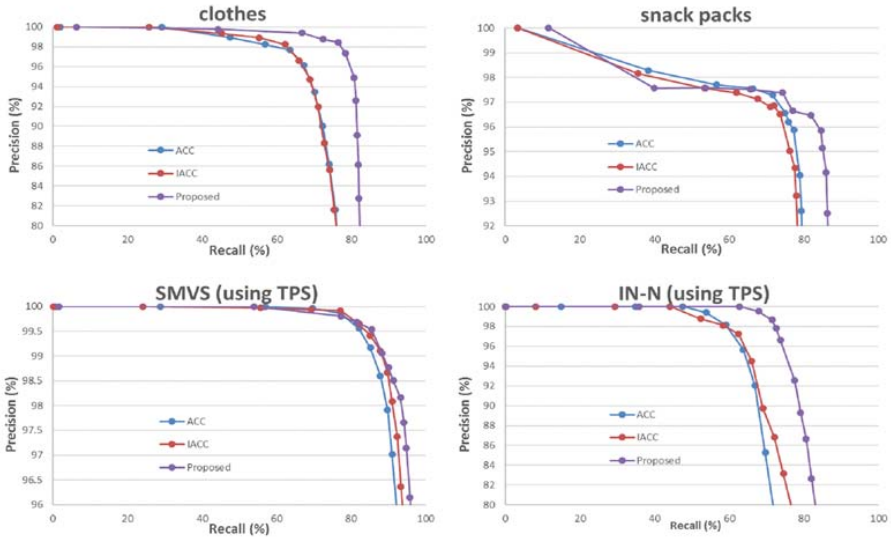


Figure 10. *Cont.*

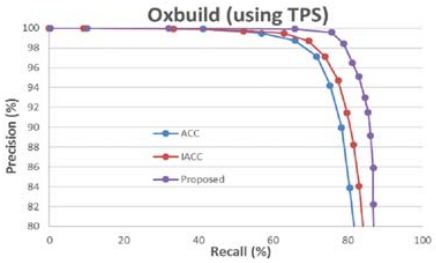


Figure 10. Recall vs. Precision curve of the proposed and other algorithms.

Tables 3–7 list the matching times for each image set. Here, the matching time means the average matching time between two images, and the unit is ms (milliseconds). The matching time was obtained by changing the value of the threshold δ_s, which is a common parameter of the three algorithms (δ_s = 1, 10, 20, 30, 40, and 50). When δ_s decreases, TPR and FPR become lower. On the other hand, when δ_s becomes larger, TPR and FPR become higher. For each algorithm, "match" and "n_match" are obtained. Here, "match" is the average matching time for the matching pairs of images, and "n_match" is the average matching time for the non-matching pairs of images. As δ_s becomes relatively large, the matching time increases, and the matching time for "match" takes longer than for "n_match". "n_match" is faster because there are relatively fewer matching pairs composed from the feature points, and there are little or no clusters composed. A comparison of the algorithms showed that the matching time of the proposed algorithm was faster than the other algorithms. In particular, for "match", it was approximately 10–70 times faster than the ACC algorithm, and approximately 2–10 times faster than the IACC algorithm. Although there was some difference depending on the image set, the proposed algorithm's matching time was the fastest.

Table 8 is a summary of the final results. The values from the table pertain to TPR (Equation (7)), FPR (Equation (8)), Accuracy (Equation (9)) and time (=matching time) in the case of δ_s where the accuracy of each algorithm is highest. Here, "time" is the total average matching time of adding "match" and "n_match" from Tables 3–7. Comprehensive examination of the results confirms that the proposed algorithm is superior to the other algorithms.

Table 3. Matching time (ms) on the "clothes" image set.

δ_s	ACC		IACC		Proposed	
	Match	n_Match	Match	n_Match	Match	n_Match
1	269.60	31.90	57.31	4.39	10.21	4.11
10	777.11	41.78	284.79	6.17	13.08	4.16
20	1113.03	64.06	436.48	8.53	18.80	4.20
30	1227.30	81.64	514.15	10.64	26.52	4.27
40	1334.33	100.00	561.21	12.36	29.65	4.34
50	1365.15	121.29	584.29	13.61	35.32	4.48

Table 4. Matching time (ms) on the "snack packs" image set.

δ_s	ACC		IACC		Proposed	
	Match	n_Match	Match	n_Match	Match	n_Match
1	62.61	6.03	10.27	5.01	7.64	4.98
10	204.05	6.43	21.05	5.08	8.11	4.97
20	231.66	6.64	23.86	5.17	8.78	4.98
30	244.62	6.80	25.71	5.24	9.38	5.03
40	252.61	6.98	26.74	5.29	10.08	5.01
50	257.75	7.09	27.59	5.32	10.49	4.99

Table 5. Matching time (ms) on the "SMVS (using TPS)" image set.

δ_s	ACC		IACC		Proposed	
	Match	n_Match	Match	n_Match	Match	n_Match
1	127.33	10.16	14.43	3.86	6.50	3.38
10	843.84	42.93	105.06	7.53	8.98	3.59
20	1063.17	69.73	142.06	10.68	10.88	3.63
30	1155.25	87.01	157.00	12.99	13.07	3.81
40	1189.76	98.73	164.42	13.53	15.42	3.95
50	1212.29	105.59	170.44	14.31	18.57	4.25

Table 6. Matching time (ms) on the "IN-N (using TPS)" image set.

δ_s	ACC		IACC		Proposed	
	Match	n_Match	match	n_Match	Match	n_Match
1	62.16	9.26	10.47	3.55	7.77	3.61
10	671.48	31.29	121.12	5.94	10.86	3.66
20	938.00	62.39	198.80	8.75	13.72	3.71
30	1072.40	85.72	240.55	10.60	16.87	3.76
40	1158.80	102.34	261.71	11.59	20.04	3.88
50	1208.94	113.75	280.67	12.63	22.93	3.87

Table 7. Matching time (ms) on the "Oxbuild (using TPS)" image set.

δ_s	ACC		IACC		Proposed	
	Match	n_Match	match	n_Match	Match	n_Match
1	115.44	22.67	34.61	9.68	21.09	7.14
10	1102.45	96.69	283.84	16.29	28.76	7.37
20	1518.45	177.52	405.74	23.83	32.11	7.42
30	1740.47	241.87	455.43	29.62	37.32	7.66
40	1826.40	278.72	486.04	32.35	44.66	7.85
50	1907.35	309.01	501.31	35.28	52.92	8.09

Table 8. Experiment results (TPR, FPR, Accuracy, and time (ms)).

Image Set	Result	RANSAC	ACC	IACC	Proposed
clothes	TPR	0.319	0.701	0.689	0.807
	FPR	0.401	0.012	0.009	0.010
	Accuracy	0.546	0.934	0.933	0.955
	time (ms)	71.98	358.22	126.21	14.77
Snack packs	TPR	0.317	0.773	0.777	0.847
	FPR	0.436	0.003	0.005	0.004
	Accuracy	0.541	0.976	0.975	0.983
	time (ms)	28.77	28.89	7.31	5.47
SMVS (using TPS)	TPR	0.983	0.923	0.923	0.948
	FPR	0.750	0.034	0.021	0.023
	Accuracy	0.585	0.946	0.954	0.963
	time (ms)	39.23	611.61	85.70	10.80
IN-N (using TPS)	TPR	0.852	0.669	0.659	0.775
	FPR	0.740	0.006	0.004	0.007
	Accuracy	0.566	0.961	0.962	0.971
	time (ms)	37.06	198.18	34.95	4.71
Oxbuild (using TPS)	TPR	0.832	0.753	0.775	0.830
	FPR	0.858	0.012	0.011	0.011
	Accuracy	0.494	0.941	0.946	0.957
	time (ms)	69.45	539.59	114.79	13.59

Figure 11 presents examples that show the matching results using the proposed algorithm, where red convex hull indicates a suitable cluster.

Figure 11. Examples of matching results using proposed algorithm.

5. Conclusions

In this paper, a new matching algorithm between images with deformable objects was proposed. A matching algorithm can be called a good algorithm if three aspects, i.e., robustness, independence, and fast matching, are excellent. Among these aspects, slow matching is the most significant weakness of conventional deformable object matching algorithms. To resolve this weakness, the speed was dramatically improved by reducing the complexity using the feature selection and BST (Binary Search Tree) clustering. The matching results were reliable because the suitability of the composed clusters is determined by the cluster verification step.

The experiment was performed using image sets with various deformable characteristics. As a result, while showing better TPR and FPR performance, compared to conventional algorithms, the proposed algorithm achieves 2–60 times faster matching speed than the conventional algorithms. Fast matching is a very important characteristic because image matching is used for content-based

image retrieval. Therefore, the algorithm proposed in this paper can be used more effectively than the conventional algorithms in deformable object-contained image retrieval.

Acknowledgments: We would like to thank the anonymous reviewers for their generous review. This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2010-0020163) and the Ministry of Science, ICT & Future Planning (2015R1C1A1A01055914).

Author Contributions: Jaehyup Jeong and Insu Won provided the main idea of this paper, designed the overall architecture of the proposed algorithm and wrote the paper; Jaehyup Jeong and HunJun Yang conducted the test data collection and designed the experiments; and Bowon Lee and Dongseork Jeong supervised the work and revised the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lowe, D.G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110. [CrossRef]
2. Bay, H.; Tuytelaars, T.; Van Gool, L. Surf: Speeded up robust features. In Proceedings of the European Conference on Computer Vision (ECCV), Graz, Austria, 7–13 May 2006; pp. 404–417.
3. Matas, J.; Chum, O.; Urban, M.; Pajdla, T. Robust wide-baseline stereo from maximally stable extremal regions. *Image Vis. Comput.* **2004**, *22*, 761–767. [CrossRef]
4. Mikolajczyk, M.; Schmid, C. Scale & affine invariant interest point detectors. *Int. J. Comput. Vis.* **2004**, *60*, 63–86.
5. Fischler, M.A.; Martin, R.C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *ACM Proc. Commun.* **1981**, *24*, 381–395. [CrossRef]
6. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the Advances in Neural Information Processing Systems (NIPS), Stateline, NV, USA, 3–8 December 2012; pp. 1097–1105.
7. Duan, L.Y.; Lin, J.; Chen, J.; Huang, T.; Gao, W. Compact Descriptors for Visual Search. *IEEE Multimed.* **2014**, *21*, 30–41. [CrossRef]
8. Chen, D.M.; Tsai, S.S.; Chandrasekhar, V.; Takacs, G. Inverted Index Compression for Scalable Image Matching. In Proceedings of the IEEE 2010 Data Compression Conference, Snowbird, UT, USA, 24–26 March 2010; p. 525.
9. Chum, O.; Matas, J.; Kittler, J. Locally optimized RANSAC. *Pattern Recognit.* **2003**, *2781*, 236–243.
10. Li, Y.; Snavely, N.; Huttenlocher, D.P. Location recognition using prioritized feature matching. In Proceedings of the European Conference on Computer Vision, Heraklion, Greece, 5–11 September 2010; pp. 791–804.
11. Na, S.; Oh, W.; Jeong, D. A Frame-Based Video Signature Method for Very Quick Video Identification and Location. *ETRI J.* **2013**, *35*, 281–291. [CrossRef]
12. Calonder, M.; Lepetit, V.; Strecha, C.; Fua, P. BRIEF: Binary Robust Independent Elementary Features. In Proceedings of the European Conference on Computer Vision (ECCV), Heraklion, Greece, 5–11 September 2010; pp. 778–792.
13. Leutenegger, S.; Chli, M.; Siegwart, R. BRISK: Binary Robust Invariant Scalable Keypoints. In Proceedings of the IEEE International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 2548–2555.
14. Alahi, A.; Ortiz, R.; Vandergheynst, P. FREAK: Fast Retina Keypoint. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; pp. 510–517.
15. Desai, A.; Lee, D.J.; Ventura, D. Matching Affine Features with the SYBA Feature Descriptor. In Proceedings of the Advances in Visual Computing, Las Vegas, NV, USA, 8–10 December 2014; pp. 448–457.
16. Fowers, S.G.; Desai, A.; Lee, D.J.; Ventura, D.; Wilde, D.K. An efficient tree-based feature descriptor and matching algorithm. *AIAA J. Aerosp. Inf. Syst.* **2014**, *11*, 596–606. [CrossRef]
17. Tran, Q.H.; Chin, T.J.; Carneiro, G.; Brown, M.S.; Suter, D. In defence of RANSAC for outlier rejection in deformable registration. In Proceedings of the European Conference on Computer Vision (ECCV), Firenze, Italy, 7–13 October 2012; pp. 274–287.
18. Pilet, J.; Lepetit, V.; Fua, P. Real-time nonrigid surface detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Miami, FL, USA, 20–25 June 2009; pp. 822–828.

19. Kettani, O.; Ramdani, F.; Tadili, B. An Agglomerative Clustering Method for Large Data Sets. *Int. J. Comput. Appl.* **2014**, *92*, 1–7. [CrossRef]
20. Zhou, F.; Torre, F.D. Factorized graph matching. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Providence, RI, USA, 16–21 June 2012; pp. 127–134.
21. Cho, M.; Lee, J.; Lee, K.M. Feature correspondence and deformable object matching via agglomerative correspondence clustering. In Proceedings of the IEEE International Conference on Computer Vision, Kyoto, Japan, 29 September–2 October 2009; pp. 1280–1287.
22. Yang, H.; Won, I.; Jeong, D. On the Improvement of Deformable Object Matching. In Proceedings of the Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV), Okinawa, Japan, 2–5 February 2014; pp. 279–282.
23. Francini, G.; Lepsøy, S.; Balestri, M. Selection of local features for visual search. *Signal Process. Image Commun.* **2013**, *28*, 311–322. [CrossRef]
24. Tsai, S.S.; Chen, D.; Takacs, G.; Chandrasekhar, V.; Vedantham, R.; Grzeszczuk, R.; Girod, B. Fast geometric re-ranking for image-based retrieval. In Proceedings of the IEEE International Conference on Image Processing, Hong Kong, China, 26–29 September 2010; pp. 1029–1032.
25. Lepsøy, S.; Francini, G.; Cordara, G.; Gusmão, P.P.B. Statistical modelling of outliers for fast visual search. In Proceedings of the IEEE International Conference on Multimedia and Expo, Barcelona, Spain, 11–15 July 2011; pp. 1–6.
26. Won, I.; Jeong, J.; Yang, H.; Kwon, J.; Jeong, D. Adaptive Image Matching Using Discrimination of Deformable Objects. *Symmetry* **2016**, *8*, 68. [CrossRef]
27. Chum, O.; Pajdla, T.; Sturm, P. The Geometric Error for Homographies. *Comput. Vis. Image Underst.* **2005**, *97*, 86–102. [CrossRef]
28. Jegou, H.; Douze, M.; Schmid, C. Hamming embedding and weak geometric consistency for large scale image search. In Proceedings of the European Conference on Computer Vision, Marseille, France, 12–18 October 2008; pp. 304–317.
29. Xie, H.; Gao, K.; Zhang, Y.; Li, J.; Liu, Y. Pairwise weak geometric consistency for large scale image search. In Proceedings of the ACM International Conference on Multimedia Retrieval, Trento, Italy, 18–20 April 2011; pp. 42–50.
30. Theodoridis, S.; Koutroumbas, K. *Pattern Recognition*, 3rd ed.; Academic Press: Cambridge, MA, USA, 2006; pp. 541–587.
31. Cormen, T.H.; Leiscrson, C.E.; Rivers, R.L.; Stein, C. *Introduction to Algorithms*, 3rd ed.; MIT Press: Cambridge, MA, USA; McGraw-Hill: New York, NY, USA, 2009; pp. 286–307.
32. Chandrasekhar, V.R.; Chen, D.M.; Tsai, S.S.; Cheung, N.; Chen, H.; Takacs, G.; Reznik, Y.; Vedantham, R.; Grzeszczuk, R.; Bach, J. The stanford mobile visual search data set. In Proceedings of the ACM Conference on Multimedia Systems, San Jose, CA, USA, 23–25 February 2011; pp. 117–122.
33. Deng, J.; Dong, W.; Socher, R. ImageNet: A Large-Scale Hierarchical Image Database. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255.
34. Philbin, J.; Chum, O.; Isard, M.; Sivic, J.; Zisserman, A. Object retrieval with large vocabularies and fast spatial matching. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Minneapolis, MN, USA, 17–22 June 2007.
35. Khan, N.; McCane, B.; Mills, S. Better than SIFT? *Mach. Vis. Appl.* **2015**, *26*, 819–836. [CrossRef]
36. Kashif, M.; Deserno, T.M.; Haak, D.; Jonas, S. Feature description with SIFT, SURF, BRIEF, BRISK, or FREAK? A general question answered for bone age assessment. *Comput. Biol. Med.* **2016**, *68*, 67–75. [CrossRef] [PubMed]