symmetry

MDPI

Article

AHiLS—An Algorithm for Establishing Hierarchy among Detected Weak Local Reflection Symmetries in Raster Images

David Podgorelec [1,*], Ivana Kolingerová [2], Luka Lovenjak [1] and Borut Žalik [1]

[1] Faculty of Electrical Engineering and Computer Science, University of Maribor, Koroška Cesta 46, SI-2000 Maribor, Slovenia; luka.lovenjak@student.um.si (L.L.); borut.zalik@um.si (B.Ž.)
[2] Department of Computer Science and Engineering, University of West Bohemia, Technická 8, 306 14 Plzen, Czech Republic; kolinger@kiv.zcu.cz
* Correspondence: david.podgorelec@um.si

Abstract: A new algorithm is presented for detecting the local weak reflection symmetries in raster images. It uses contours extracted from the segmented image. A convex hull is constructed on the contours, and so-called anchor points are placed on it. The bundles of symmetry line candidates are placed in these points. Each line splits the plane into two open half-planes and arranges the contours into three sets: the first contains the contours pierced by the considered line, while the second and the third include the contours located in one or the other half-plane. The contours are then checked for the reflection symmetry. This means looking for self-symmetries in the first set, and symmetric pairs with one contour in the second set and one contour in the third set. The line which is evaluated as the best symmetry line is selected. After that, the symmetric contours are removed from sets two and three. The remaining contours are then checked again for symmetry. A multi-branch tree representing the hierarchy of the detected local symmetries is the result of the algorithm.

Keywords: computer science; algorithm; computer vision; computational geometry; shape features

check for updates

Citation: Podgorelec, D.; Kolingerová, I.; Lovenjak, L.; Žalik, B. AHiLS—An Algorithm for Establishing Hierarchy among Detected Weak Local Reflection Symmetries in Raster Images. *Symmetry* **2024**, *16*, 442. https://doi.org/10.3390/sym16040442

Academic Editors: Changxin Gao and Lorentz Jäntschi

Received: 16 January 2024
Revised: 8 March 2024
Accepted: 1 April 2024
Published: 6 April 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Symmetry is a fundamental and versatile concept with applications in mathematics [1–3], natural sciences [4], architecture [5], arts [6], engineering [7–9], and elsewhere [10,11]. Through evolution, symmetry perception has become an important integral part of an individual's perceptual organisation process [12]. It is mostly understood as a positive and desirable property, which has been recognised in studies of mating and food choice habits of many animal species already [13]. In humans, this concept is further generalised to broader perceptions of aesthetics, health, safety, and stability. Symmetry applies to various abstractions, but is most often associated with visual perception, which is characterised by the detection and interpretation of distances (depth and size) and colours. Within the human's visual system, symmetries facilitate the segmentation of the seen scene, object analysis and representation [14]. Unsurprisingly, computer applications of symmetries, once they have been detected, also mostly address these tasks. In computing, the symmetry phenomenon may play an important role in image processing [15], computer vision and computer graphics [16], cryptography [17], and geometric modelling [18]. However, in contrast to natural, almost self-evident symmetry perception processes in humans and several animal species, symmetry detection is anything but a simple task for a computer [19].

An object or a system (a set of points in our case) is symmetric if there is a transformation, such as a translation, rotation, reflection, or a combination of these, that maps onto itself [20]. Formally, if $\mathcal{P}$ denotes a set of points and $\mathbf{T}$ some geometric transformation, then $\mathcal{P}$ is symmetric when

$$\forall p \in \mathcal{P} : \mathbf{T}(p) \in \mathcal{P}. \quad (1)$$

When $\mathcal{P}$ corresponds to the entire considered set of points, then Equation (1) represents global symmetry. On the other hand, local symmetry addresses a subset of the input set of points. It is also sometimes called partial symmetry [14].

In practice, however, Equation (1) is rarely satisfied exactly for all elements of $\mathcal{P}$. Any observation of the physical world is subject to error, whether by human vision or by remote sensing. As a consequence, we can only expect an exact match between two points in a symmetric pair in the case of synthetic or rounded data. Instead of perfect (exact, strong) symmetry, we thus consider weak (or approximate) symmetry [21,22]. This can be described formally by Equation (2).

$$\forall p \in \mathcal{P}, \exists p' \in \mathcal{P} : (\mathbf{T}(p') \in \mathcal{P}) \wedge (\|\mathbf{T}(p) - \mathbf{T}(p')\| < \epsilon), \tag{2}$$

where ϵ is a small positive tolerance value. Note that Equation (2) does not require $\mathbf{T}(p) \in \mathcal{P}$.

The distinction between weak and perfect symmetry concerns both global and local symmetry. Local symmetries can contain patterns and repetitions that are often not identified easily by humans, in contrast to global symmetry, where humans are extremely skilful at its detection [19,23]. Similarly, humans can easily detect and interpret local symmetries of isolated objects and their exposed parts in a similar way to global symmetries. On the other hand, no symmetry is trivial for a computer, but computer algorithms can use a lot of processing power even in situations with a huge number of less noticeable symmetries, which can also intersect, nest inside each other, or extend over individual tiny details. The algorithm from [14], for example, detects more than 8500 local reflection symmetries in a complex 3D scene of floor, buildings, and trees with more than 270,000 LiDAR points in 3 min by using a single-threaded CPU processing on a regular personal computer. In a slightly simpler scene with 125,000 points and no trees, it finds almost 25,000 symmetries in less than half a minute, which of course drastically exceeds the capabilities of a human. However, to use the detected symmetries in, e.g., scene segmentation or object analysis, it is not enough for the algorithm to find as many symmetries as possible, but it also needs to structure them appropriately. This is precisely the aim of this paper. The proposed algorithm for detecting and establishing a hierarchy of the weak local symmetries in raster images (the acronym AHiLS is used from here on), thus, also determines hierarchical relationships among the found weak local reflection symmetries, if they exist. The aim of this paper is not only to find weak local symmetries, but also hierarchical relationships among them, if they exist. We consider weak symmetry when talking about symmetry from here on.

To our knowledge, no methods in the past have addressed the same challenges as the proposed approach. The main novelties introduced by AHiLS are the following:

- a new algorithm for weak local symmetry detection in raster images;
- performing symmetry detection on extracting contours previously extracted from the raster image;
- using a convex hull of contours to determine the positions of potential symmetry line candidates;
- representing hierarchical dependencies by a multi-branch tree for further processing.

This paper consists of five Sections. Section 2 gives an overview of the most similar previous works. The proposed method is described in Section 3 and its results are given in Section 4. A discussion and suggestions for future work are given in Section 5.

2. Related Works

Most of the previous work concentrates on perfect symmetry. Some methods compare either two halves of the contours or skeletons of the object [24–27], or analyse halftone images [28]. Zahn and Roskies [29] and Yip at al. [30] find symmetries in geometric objects segmented from high-quality images. However, in real life, images can be undersampled, which introduces additional errors. The algorithm by Loy and Eklundh [31] extracts

feature points first from an image using SIFT transform [32], which is invariant on scaling and rotation. SIFT also constructs a descriptor for each feature point. A reflected image is generated after that, and SIFT is applied to it too. A new set of feature points and their descriptors are obtained in this way. These descriptors are reflected after that, and faced for matching against the descriptors obtained from the original, non-reflected image. The matched descriptors form pairs. These pairs are used to define the symmetry lines, which are accumulated into the Hough-style voting space [33]. The most dominant symmetry lines are then selected. O'Mara and Owens in [34] present a method for the detection of the dominant plane of symmetry in medical images of an arbitrary dimension with the aim of measuring the degree of detected symmetries. Sun and Si published a reflectional symmetry detection algorithm for greyscale images [35]. Axes of symmetry for an object in the image are detected on the basis of the gradient information of the image and the Fourier transformation. The algorithm described in [36] generates the candidates for the axes of symmetry and then searches for the optimal axis by reflecting the candidates against these axes. Hauagge and Snavely [37] extracted local features from images of architectural scenes, based on local symmetries' detection and extraction. They matched pairs of photos of urban scenes according to these features. The features were based on the measures of the local reflection and rotational symmetries, computed by the so-called local image operations.

Van Gool et al., in [38], discuss the detection and usage of symmetry in planar shapes. Symmetry is interpreted as repeated, coplanar shape fragments. These fragments are tested on whether they are symmetric. The so-called arc length space is proposed, in which the symmetric shape fragments correspond to straight-line segments, which are detected and analysed easily. The algorithm proposed by Derrode and Ghorbel [39] uses the analytical Fourier–Mellin transform to assess motion parameters between grey-level objects having the same shape with distinct scales and orientations. Both rotational and reflection symmetries can be detected and estimated in objects. Karkischenko and Mnukhin [28] propose an algorithm for rotational symmetry detection in grey-level digital images, based on the concept of Gaussian fields and a special log-polar representation of digital images. Gnutti et al. [40] address the problem of reflection symmetry detection. A metric is designed that extracts subsets of consistently oriented candidate segments. These segments are ranked on the basis of the surrounding gradient orientation specularity. The used operations are considered to be related to the way the human brain detects symmetry.

In comparison to the existing approaches, the proposed algorithm handles more general types of input data and is simple to understand. Although it is demonstrated on raster images, it can also be applied directly to geometric objects, whose borders are represented by contours. In addition, it detects the hierarchies of the local symmetries and returns this information explicitly in the multi-branch tree. Hierarchical dependencies between symmetries have been considered less often. The algorithm in [41] combines a continuous symmetry measure with a multiresolution scheme to detect hierarchical symmetry and almost-symmetrical patterns. Ref. [42] presents a symmetry hierarchy constructed from the initial graph via recursive graph contraction, which either groups parts by symmetry or assembles connected sets of parts. The method is designed for meshed man-made models. In [19], 2D or 3D objects with or without holes are found using the algorithm, where a uniform grid and a symmetry estimation function are utilised. Li et al. [43] predict poses of a 3D assembly by using a single 2D image with the assembly in a considered pose and a set of point clouds of individual parts of the assembly. The reflection symmetry is an important, but is not the only feature used to describe the poses. Paschalidou et al. [44] address a similar problem of learning and predicting an unsupervised decomposition of a 3D object, extracted from a single 2D image, into a hierarchical structure of parts. Xue et al. [45] proposed a novel unsupervised approach to process urban LiDAR point clouds to a hierarchy of objects based on their characteristic symmetric cross-sections. The method, called Clustering Of Symmetric Cross-sections of Objects (COSCO), was primarily used to detect and classify cars, which are then replaced by free 3D car models in

a digital twin city. Villanueva et al. [46] use symmetries for compression of sparse voxel grids. Originally, two subtrees could be merged on the basis of similarity or connectivity, which they extended to the use of reflection symmetries as well. In this manner, the concept of Sparse Voxel Directed Acyclic Graphs (SVDAGs) was upgraded into the Symmetry-Aware Sparse Voxel Directed Acyclic Graphs (SSVDAGs), which were further analysed by Madoš et al. [47]. Although all these methods use a similar concept of symmetry-aware hierarchical organisation of spatial data, they were designed for different purposes, and thus, none of them is directly comparable to AHiLS. Some allow only a hierarchical division perpendicular to the symmetry axis or plane [19,45], some are designed from the bottom up and use other principles of grouping subtrees together with symmetry [42–44], and some are restricted to global symmetry detection [19].

3. Materials and Methods

In the following section, the elements of $\mathcal{P}$ correspond to pixels $p_{x,y}$ in a colour raster image, i.e., $p_{x,y} \in \mathcal{P}$, $0 \leq x < X$, $0 \leq y < Y$, where X and Y define the image resolution, while the transformation $\mathbf{T}$ is restricted to the reflection/mirroring. AHiLS executes several steps to determine the hierarchy among the weak local reflection symmetries. The pseudocodes in Algorithms 1 and 2 give the sequence of these steps, which are explained afterwards.

Algorithm 1 The main algorithm of the AHiLS

```
1:  function AHiLS(P)
2:  ▷ Function accepts a colour image P and returns the multi-branch tree T representing
    the weak local symmetries.
3:      InitialiseParameters(R_c, t, b, d);
4:      S = ApplyColourSegmentation(R_c);
5:      S = Filter(S);
6:      C = ExtractsContours(S);
7:      level = 0;
8:      T = NIL;
9:      repeat
10:         (Γ, T) = DFS(C, level, b, d, T);
11:         C = C − Γ;
12:     until Γ = ∅;
13:     return T;
14: end function
```

AHiLS accepts $\mathcal{P}$ (for example, the image shown in Figure 1a) and performs the initialisation steps within lines 3–8 in Algorithm 1 as follows:

(**a**) (**b**) (**c**)

Figure 1. (**a**) Input raster image (credit: Paul Reeves Photography/https://www.shutterstock.com, accessed on 7 March 2024); (**b**) segments of interest; (**c**) detected contours.

1. The necessary parameters, explained in the following, are set in line 3.
2. $\mathcal{P}$ is segmented based on colours (line 4 in Algorithm 1), where a user specifies the range R_c of the desired colours. Pixels with colours outside R_c are set to black. The segmented image $\mathcal{S}$, suitable for further processing, is obtained in this way (see Figure 1b). The segment consists of a set of four connected pixels and should be large

enough. In our implementation, the segment is accepted when it contains at least 1% of all pixels in $\mathcal{S}$.

3. $\mathcal{S}$ is converted into greyscale, filtered to reduce the noise, and smoothed in line 5. For this, Gaussian blur and morphological erosion are applied [48].
4. The set $\mathcal{C}$ of contours C_i, $0 \leq i < |\mathcal{C}|$, bordering the segments, is extracted from $\mathcal{S}$ using a 5×5 kernel [48] in line 6. Fifty contours were obtained in our case (Figure 1c).
5. The level of the hierarchy and the multi-branch tree $\mathcal{T}$ are initialised in lines 7 and 8.

Algorithm 2 A depth-first search investigation of contours

1: **function** DFS($\mathcal{C}$, *level*, b, d, Γ, $\mathcal{T}$)
2: H = DetermineConvexHull($\mathcal{C}$);
3: $\mathcal{A}$ = PlaceAnchorPointsOnHull(H, d);
4: $\Gamma = \emptyset$; $\Gamma_A = \emptyset$; $\Gamma_B = \emptyset$; $\Gamma_P = \emptyset$;
5: **for** $a \leftarrow 0$ to $|\mathcal{A}| - 1$ **do** ▷ for all bundles of lines
6: $\mathcal{L}$ = PlaceBundleOfLinesInAnchorPoint(p_a, b);
7: **for** $s \leftarrow 0$ **to** $b - 1$ **do** ▷ for all lines in a bundle
8: ($\mathcal{C}_A$, $\mathcal{C}_B$, $\mathcal{C}_P$) = ClassifyContours($\mathcal{C}$, $l_s \in \mathcal{L}$);
9: **if** $\mathcal{C}_A \neq \emptyset$ **and** $\mathcal{C}_B \neq \emptyset$ **then**
10: $\mathcal{C}_B^R$ = Reflect($\mathcal{C}_B$, l_s);
11: (*ListOfPairs*, *ListOfCoverages*) = FindSymmetricPairs($\mathcal{C}_A$, $\mathcal{C}_B^R$, t);
12: $\Gamma_P = \Gamma_P \cup$ StorePairsAndCoverages(*ListOfPairs*, *ListOfCoverages*);
13: **end if**
14: **end for** ▷ for all lines in a bundle
15: **for** $q \leftarrow 0$ **to** $|\mathcal{C}_P| - 1$ **do** ▷ for all contours in $\mathcal{C}_P$
16: c_q = CalculateSelfSymmetricCoverage($C_q \in \mathcal{C}_P$);
17: **if** $c_q > t$ **then**
18: $(\Gamma, \mathcal{T})$ = StoreSelfSymContourAndAddItToTree(c_q, l_s, Γ, $\mathcal{T}$);
19: **end if**
20: **end for** ▷ for all contours in $\mathcal{C}_P$
21: **end for** ▷ for all bundles of lines
22: l_{best} = SelectBestLineFromStorage(Γ_P);
23: **if** $l_{best} \neq NIL$ **then**
24: $(\Gamma, \mathcal{T})$ = FormPairsAndAddThemToTree(l_{best}, *level*, Γ, $\mathcal{T}$);
25: ($\mathcal{C}_A$, $\mathcal{C}_B$, $\mathcal{C}_P$) = ClassifyContours($\mathcal{C}$, l_{best});
26: $(\Gamma_A, \mathcal{T})$ = DFS($\mathcal{C}_A$, *level* + 1, b, d, $\mathcal{T}$);
27: $(\Gamma_B, \mathcal{T})$ = DFS($\mathcal{C}_B$, *level* + 1, b, d, $\mathcal{T}$);
28: **end if**
29: **return** $(\Gamma \cup \Gamma_A \cup \Gamma_B, \mathcal{T})$
30: **end function**

The algorithm enters the main loop after that. Function DFS (line 10) finds in the depth-first search order the symmetric contours and returns them in set Γ, together with the constructed multi-branch tree $\mathcal{T}$. The found symmetric pairs of contours are removed from $\mathcal{C}$ in line 11 and the whole process repeats with the reduced set of contours. The loop terminates when DFS returns an empty Γ. The multi-branch tree is returned at the end (line 13).

The core of AHiLS is realised in the recursive function DFS, shown in Algorithm 2. Figure 2a is used to clarify the explanation.

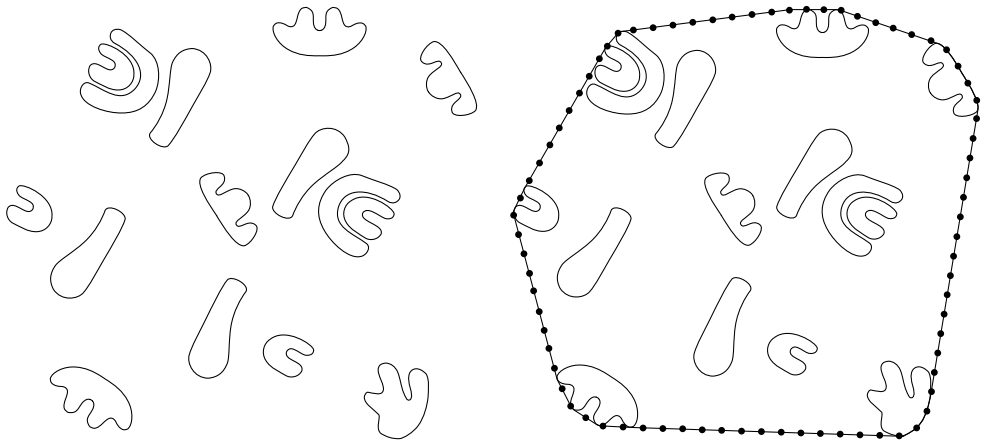

Figure 2. (**a**) Set of contours and (**b**) their convex hull with the anchor points.

1. Convex hull H [49] is constructed from contours C_i, $0 \leq i < |\mathcal{C}|$ at first (line 2 in Algorithm 2). The set $\mathcal{A}$ of so-called anchor points p_a is placed along the convex hull's edges uniformly (line 3), as shown in Figure 2b. The distance d between the consecutive anchor points on H is specified heuristically (in our case $d = 5$ was used as a default) or entered by the user. The number of anchor points is reduced drastically by placing them on H instead of, for example, distributing them uniformly inside the whole $\mathcal{S}$.
2. A bundle $\mathcal{L}$ of potential symmetry lines l_s, $0 \leq s < 180/b$, $b \in \{1, 2, 4, 5, 10\}$, with slopes $\alpha = b \cdot s$, is laid through p_a (see Figure 3a) in line 6. b is set by default to 5, resulting in $|\mathcal{L}| = 36$ lines.

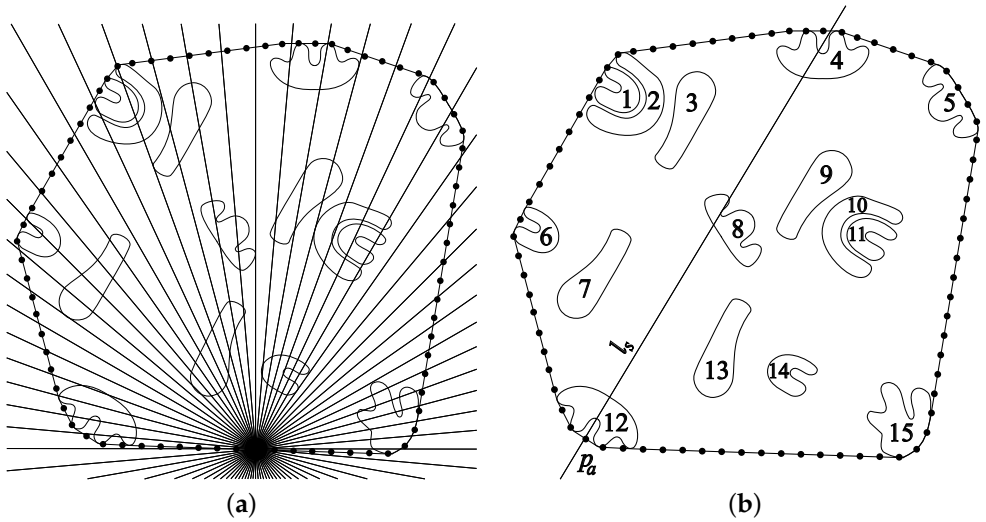

Figure 3. (**a**) Bundle of potential lines of symmetry placed in an anchor point; (**b**) considered line of symmetry l_s passes through an anchor point $p_a \in \mathcal{A}$ and splits $\mathcal{C}$ into three sets: $\mathcal{C}_A = \{1, 2, 3, 6, 7\}$, $\mathcal{C}_B = \{5, 9, 10, 11, 13, 14, 15\}$, and $\mathcal{C}_P = \{4, 8, 12\}$.

3. The contours in $\mathcal{C}$ are classified according to $l_s \in \mathcal{L}$. Namely, l_s determines two half-planes, and (in some cases) splits $\mathcal{C}$ into three sets (see Figure 3b):
   - set $\mathcal{C}_A$ contains contours being located completely in the first half-plane;
   - set $\mathcal{C}_B$ includes contours from the opposite half-plane;
   - set $\mathcal{C}_P$ stores those contours that are pierced by l_s.

Those sets are obtained by the function in line 8. If one of the sets $\mathcal{C}_A$ or $\mathcal{C}_B$ is empty, l_s is not the line of symmetry for any pairs of contours. Otherwise, $\mathcal{C}_A$ and $\mathcal{C}_B$ must be tested additionally as follows.

- Contours from $\mathcal{C}_B$ are reflected over l_s to obtain set $\mathcal{C}_B^R$ in line 10 (see Figure 4a).
- The similarity of the contours in $\mathcal{C}_A$ and $\mathcal{C}_B^R$ is checked. The coverage percentage of the two considered contours is calculated for this. The two contours are paired and returned in the *ListOfPairs* in line 11 if the coverage percentage is greater than the threshold t. The percentage of coverage is returned in the list *ListOfCoverages*. The threshold t is set at 90% in our implementation. However, t can be determined interactively by the user. Its effect is demonstrated later in Section 4.
- This information is then stored in a temporal storage Γ_P in line 12 of Algorithm 2.

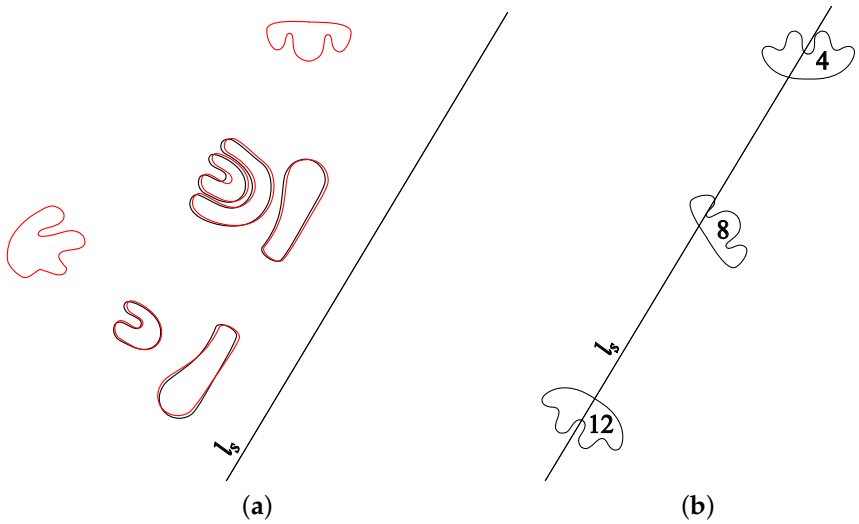

Figure 4. (**a**) Contours from $\mathcal{C}_B$ (plotted in red) are reflected according to l_s and checked for matching with the contours from $\mathcal{C}_A$. (**b**) Contours in $\mathcal{C}_C$ are checked for self-symmetry with regards to l_s.

4. The algorithm selects the best line of symmetry l_{best} in line 22 after processing all l_s candidates from all $p_a \in \mathcal{A}$. The criterion for selection is the highest number of reflection contour pairs determined by this line. The line with the highest percentage of contour coverage is selected if more lines have the same number of pairs.
5. The contours in $\mathcal{C}_P$ are checked for self-symmetry (see Figure 4b) in lines 15–20. Those that are symmetric are stored in Γ and inserted into $\mathcal{T}$.
6. If l_{best} exits, the DFS is called recursively with sets $\mathcal{C}_A$ and $\mathcal{C}_B$ in lines 26 and 27 in Algorithm 2.
7. The information about the found symmetries stored in Γ is, together with the multi-branch tree, returned to the main part of the algorithm at the end. The multi-branch tree for our example is shown in Figure 5.

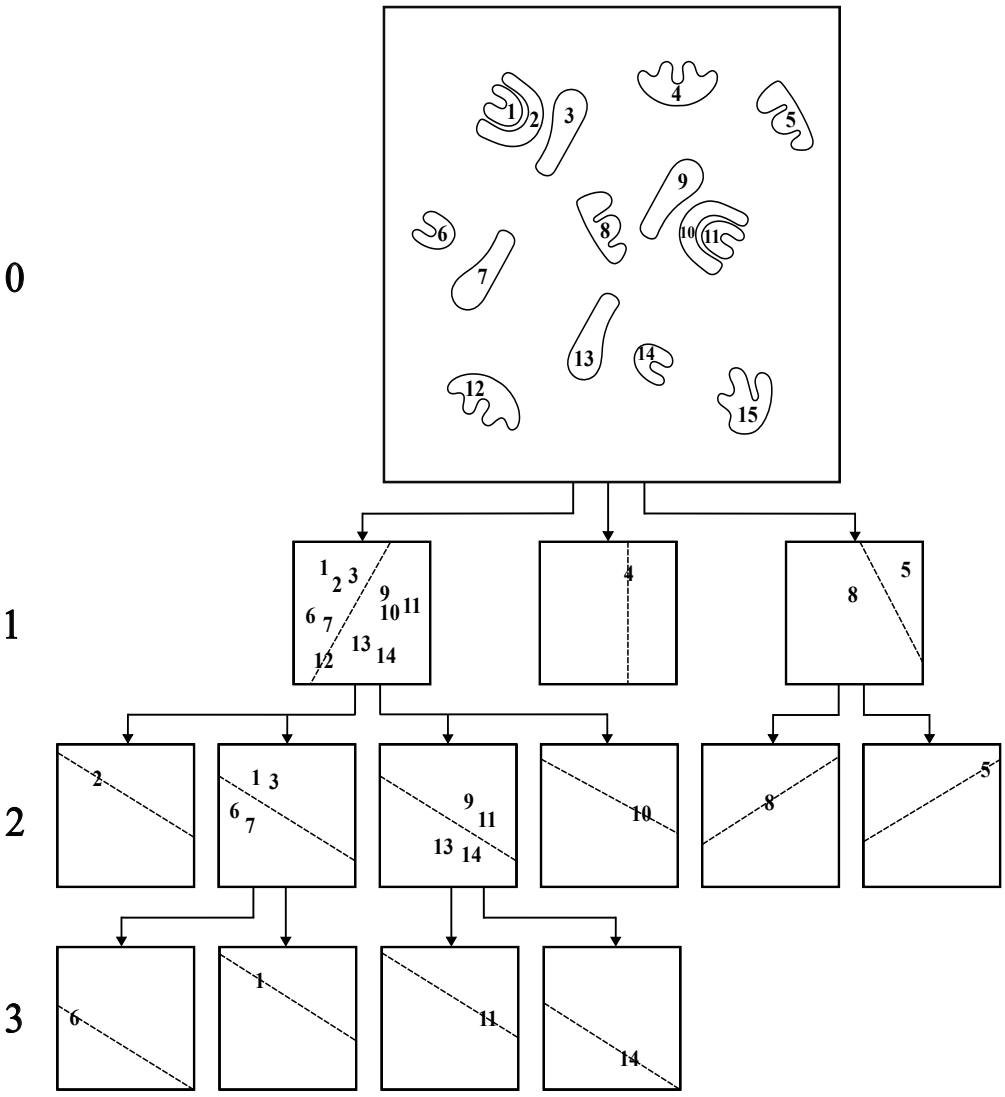

Figure 5. The obtained multi-branch tree of detected local symmetries.

4. Results

Figure 6 shows AHiLS's results when varying the parameter t, which represents the coverage percentage. This is the most important of the three user-defined parameters in AHiLS. Setting it would require intensive study from the human-sense perspective to obtain the most expected results. Namely, humans are not very skilful at determining local symmetries in contrast to global ones. The other two parameters, b and d, were set as a compromise between AHiLS's accuracy and its spent CPU time, as they directly affect the number of symmetry line candidates in each bundle and the number of anchor points along the convex hull, respectively. They are set to $b = 2$ and $d = 5$ in the test example in Figure 6. Only one symmetry line was detected when $t = 50\%$. The symmetric contours are plotted in red while the others are in grey (see Figure 6a). However, there were more lines of symmetry discovered by increasing t, as seen in Figure 6b. Different colours are used to show which contours and axes of symmetry belong together. In addition, some contours were also detected as being self-symmetric when t was further increased (Figure 6c,d).

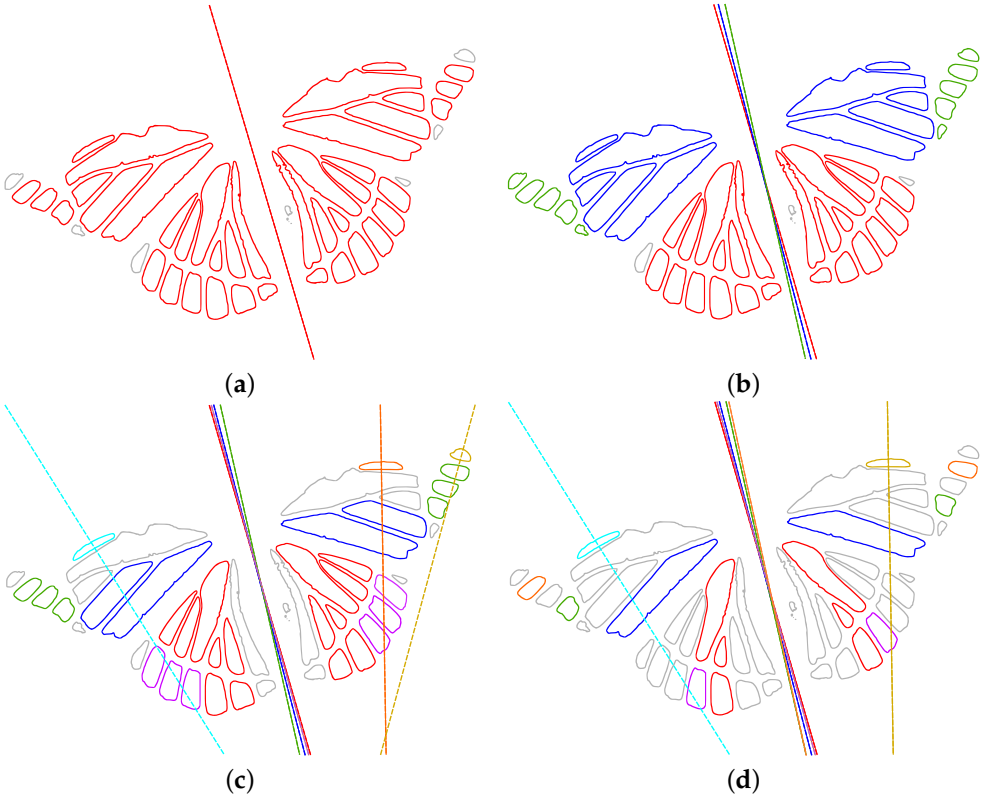

Figure 6. The found symmetric contours are plotted with the same colour in regards to varying t: (**a**) 50%, (**b**) 70%, (**c**) 80%, (**d**) 90%.

Figure 7 shows the multi-branch tree obtained for $t = 70\%$. Note that the symmetry lines at level 1 are similar, but not exactly the same. Consequently, three branches were obtained instead of a single one, which corresponds to Figure 6b.

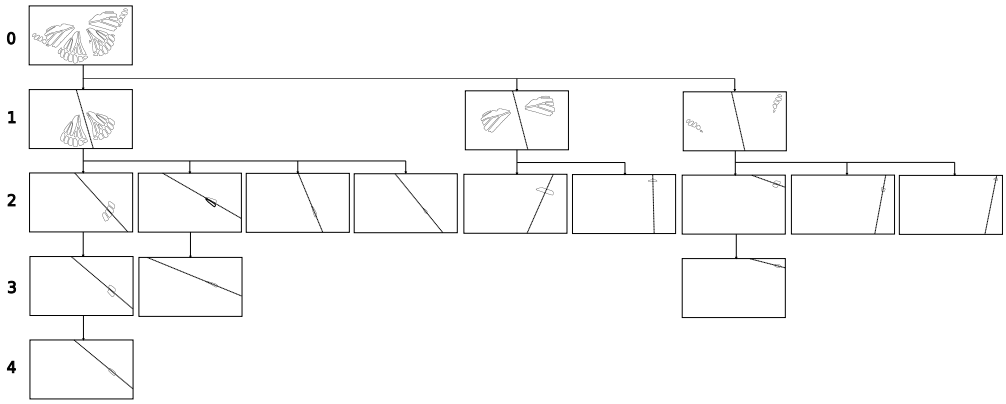

Figure 7. Butterfly example reflection hierarchy.

Finally, a comparison was made with the method proposed by Loy and Eklundh [31]. As already mentioned, we have not found any comparable method that would hierarchically organise the detected symmetries in a 2D image in a similar way as AHiLS. Therefore, we were looking for a method that would at least pre-segment and filter the image, or otherwise incorporate some semantic relations in the symmetry detection rather than pixel colours alone. Thus, the Loy and Eklundh (L&E in continuation) method was chosen, which bases the detection on a prior identification of feature points. However, this approach does not filter out the background, which often appears dominant in at least one of the detected symmetries. Since the L&E method does not build a hierarchical structure but only finds a selected number of the strongest symmetries, we used only the topmost level in the

multi-branch tree in AHiLS to make the comparison fairer. The numbers of symmetries in the L&E method were set to the number of tree nodes considered in AHiLS (increased by 1 for the strongest background symmetry). Different colours in Figure 8b,d,f are used to bind the symmetry lines to the corresponding contours.

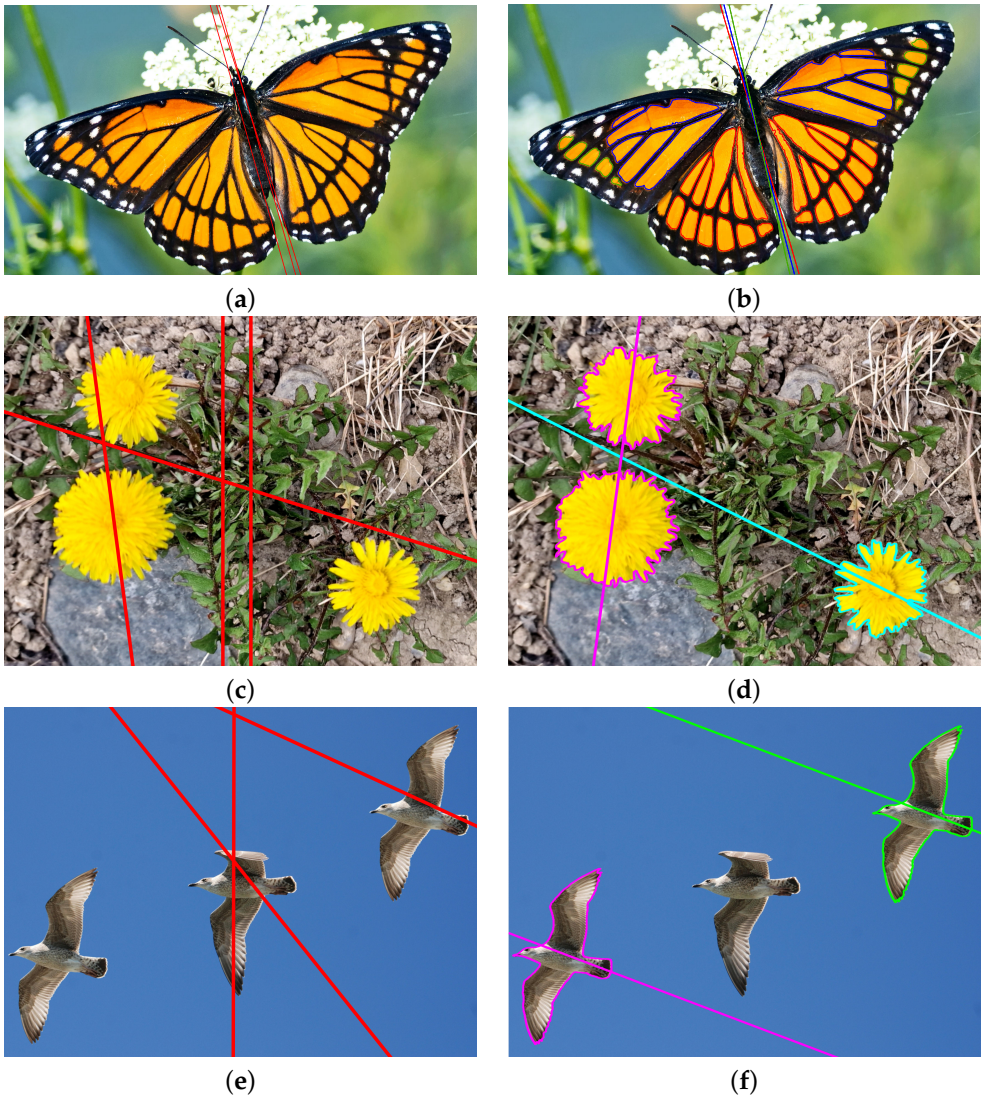

Figure 8. Results of L&E [31] (**a**,**c**,**e**) and AHiLS (**b**,**d**,**f**) methods, where $t = 70\%$, $b = 2$, and $d = 5$. (credit: Paul Reeves Photography/https://www.shutterstock.com, accessed on 7 March 2024, for (**a**,**b**), and https://www.pexels.com/, accessed on 7 March 2024, for (**e**,**f**)).

The symmetries found in the butterfly example (Figure 8a,b) are quite similar for both methods. Here, the background is highly asymmetrical (predominantly blue in the left half of the image and green in the right half), so the strongest symmetries are dominated by parts of the butterfly even in the L&E method. In the case of dandelion flowers, however, the L&E algorithm detected two strong symmetries with the vertical axes near the centre of Figure 8c. The left one refers to the symmetry of the green leaves, while the right one additionally takes into account the soil. The least steep symmetry axis also refers to the green leaves and the soil, while the fourth, steeper, axis roughly bisects the flower on the bottom left and the stone below it. On the other hand, AHiLS, where R_c was restricted to shades of yellow, first found symmetry through the two flowers on the left, indicated by magenta, and then found the self-symmetry of the third flower, indicated by cyan in Figure 8d. If AHiLS was not a hierarchical algorithm, it would also find the symmetries through the remaining two pairs of the trio of flowers. In the third example, showing three

seagulls flying under blue sky, the L&E algorithm detected two strong symmetries nearly through the centre of Figure 8e, where the vertical one corresponds to the background, while the oblique one considers both the sky and the birds. The third symmetry axis considers the rightmost seagull and its local surroundings. Note that this and the leftmost seagull are two identical copies, but the local symmetry of the latter is only at the tenth position due to the higher influence of the middle bird. As expected, AHiLS found here the local self-symmetries of the two identical seagull copies, highlighted in magenta and light green in Figure 8f. It should be noted that the light green axis is not identical to the third red axis of symmetry from the L&E algorithm, as the latter is also affected by the adjacent seagull and the background.

As can be seen, AHiLS not only found more meaningful lines of symmetries, but it was also able to define the exact regions of the symmetry in $\mathcal{S}$. However, the price for this is a much slower performance. The L&E method needs 0.69 s, 1.19 s, and 0.40 s for the results in Figure 8a, Figure 8c, and Figure 8e, respectively. On the other hand, AHiLS needs 825.22 s, 189.13 s, and 166.88 s for the results in Figure 8b,d,f. Deeper levels of the tree in principle take less time, as we no longer have to deal with objects pierced by the axis of symmetry at a higher level. Also, the convex hulls are becoming shorter, which means far fewer anchor points. Nevertheless, run times are increased by a further factor of 2 to 3 when DFS is performed recursively.

The tests were run on a laptop computer with an 11th Gen Intel(R) Core(TM) i7-1165G7 processor @ 2.80 GHz, 2803 MHz, with four cores, eight logical processors, and with 16.0 GB RAM, under the Microsoft Windows 11 Home operating system. The software was written in Python. The resolution of the butterfly image was 1600×915 pixels, while the other two test images had resolutions of 1200×900 pixels.

5. Conclusions

This paper introduces a novel method for determining weak local reflection symmetries and their hierarchical dependencies in raster images and returns the information about them in a multi-branch tree. Although the method was tested on raster images, it can also be applied without any changes to non-rasterised cases, when the geometry of the presented shapes is described by their boundaries (by contours in AHiLS terminology). The obtained multi-branch tree allows a comprehensive analysis of the detected symmetries in the later post-processing phases.

In addition to the original idea to hierarchically detect and organise symmetries from top to bottom, the main advantage of AHiLS is its flexibility. Three user-defined parameters t, b, and d play an important role in the symmetry detection. Furthermore, a user can also set the colour range R_c for filtering and segmentation of the image prior to symmetry detection. However, with so many parameters, the challenge for future work is to automatically recommend their values based on image and user characteristics. An obvious weakness of AHiLS is its slowness. The method can, however, be parallelised easily. A divide and conquer approach, as suggested in [19], should thus also be tried within AHiLS. Furthermore, calculation of coverages could be avoided if the two contours already differ in length by more than the set threshold t. Note that a coverage between the same pair of contours is calculated separately for every symmetry line candidate through each anchor point. Testing the anchor points is also highly time consuming, therefore a smarter anchor point selection could significantly improve the performance. Finally, the AHiLS methodology should also be adapted for other types of symmetries, first of all for rotational symmetry.

Author Contributions: Conceptualisation, B.Ž. and I.K.; methodology, B.Ž. and L.L.; software, L.L.; validation, B.Ž. and D.P.; investigation, B.Ž., I.K., L.L. and D.P.; resources, I.K.; writing—original draft preparation, L.L. and I.K.; writing—review and editing, B.Ž. and D.P.; visualisation, L.L.; supervision, B.Ž.; project administration, I.K.; funding acquisition, I.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Slovene Research and Innovation Agency under Research Project N2-0181 and Research Programme P2-0041, and the Czech Science Foundation under Research Project 21-08009K.

Data Availability Statement: No new data were created or analysed in this study. Data sharing is not applicable to this article.

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

Abbreviations

The following abbreviations and symbols are used in this manuscript:

α	Slope angle of a considered l_s, i.e., $\alpha = b \cdot s$, $0 \leq s < 180^\circ/b$
Γ	Set of symmetric contours
Γ_A	Output set of symmetric contours from the recursive call DFS($\mathcal{C}_A$,...)
Γ_B	Output set of symmetric contours from the recursive call DFS($\mathcal{C}_B$,...)
Γ_P	Temporal storage of symmetric pairs and coverage values
ϵ	Small positive tolerance value
a	Loop index for anchor points
A	Set of anchor points along the convex hull H
AHiLS	Algorithm for establishing hierarchy of weak local symmetries in raster images
b	User parameter for setting the slope step ($180^\circ/b$) for lines l_s in a bundle
$\mathcal{C}$	The set of contours (borders of segments from $\mathcal{S}$; formal parameter of DFS)
$\mathcal{C}_A$	Set of contours completely in one half-plane defined by line l_s
$\mathcal{C}_B$	Set of contours from the opposite half-plane to the one containg $\mathcal{C}_A$
C_i	Contours from the set $\mathcal{C}$, $0 \leq i < \lvert\mathcal{C}\rvert$
COSCO	Clustering of Symmetric Cross-Sections of Objects
$\mathcal{C}_P$	Set of contours being pierced by l_s
CPU	Central processing unit
c_q	Self-symmetry coverage percentage of the contour C_q (line 16 of Algorithm 2)
C_q	Contour from $\mathcal{C}_P$, $0 \leq q < \lvert\mathcal{C}_P\rvert$ (line 16 of Algorithm 2)
$\mathcal{C}_B^R$	Set of contours obtained by reflecting $\mathcal{C}_B$ over l_s
d	User parameter for the distance step between two consecutive anchor points
DFS	Depth-first search
H	Convex hull created from the contours of $\mathcal{C}$ (input parameter of DFS)
$\mathcal{L}$	Bundle of potential symmetry lines through a considered anchor point
L&E method	The Loy and Eklundh method
level	Index of the considered level in the multi-branch tree $\mathcal{T}$
l_{best}	Best symmetry line of all l_s from all $p_a \in A$
LiDAR	Light detection and ranging
ListOfPairs	List of symmetric pairs of contours from $\mathcal{C}_A$ and $\mathcal{C}_B$
ListOfCoverages	List of coverage percentage values of pairs of contours from $\mathcal{C}_A$ and $\mathcal{C}_B^R$
l_s	Line in a bundle through a considered p_a, $0 \leq s < 180^\circ/b$
p	Point
$\mathcal{P}$	Set of points (raster image)
p'	Point (close to p in the weak symmetry definition)
p_a	Anchor point
$p_{x,y}$	Pixel in the x-th column and y-th row of the raster image $\mathcal{P}$
q	Loop index for contours in $\mathcal{C}_P$
R_c	Range of colours
s	Loop index for lines in a bundle
$\mathcal{S}$	Segmented (and filtered) image
SIFT	Scale-invariant feature transform
SSVDAG	Symmetry-Aware Sparse Voxel Directed Acyclic Graph
SVDAG	Sparse Voxel Directed Acyclic Graph
t	User-defined coverage percentage threshold for pairing two contours
T	Geometric transformation
$\mathcal{T}$	Multi-branch tree of week local symmetries
X, Y	Image resolution (numbers of columns and rows)

References

1. Barker, W.H.; Howe, R. *Continuous Symmetry: From Euclid to Klein*; American Mathematical Society: Providence, RI, USA, 2007.
2. Jäntschi, L.; Bolboacã, S.D. *Symmetry in Applied Mathematics*; MDPI: Basel, Switzerland, 2020.
3. Dias G.; Liberti, L. Exploiting symmetries in mathematical programming via orbital independence. *Ann. Oper. Res.* **2021**, *298*, 1–34. [CrossRef]
4. Evans, C.S; Wenderoth, P.; Cheng, K. Detection of Bilateral Symmetry in Complex Biological Images. *Perception* **2000**, *29*, 31–42. [CrossRef] [PubMed]
5. Mehaffy, M.W. The Impacts of Symmetry in Architecture and Urbanism: Toward a New Research Agenda. *Buildings* **2020**, *10*, 249. [CrossRef]
6. McManus, I.C. Symmetry and Asymmetry in Aesthetics and the Arts. *Eur Rev.* **2005**, *13*, 157–180. [CrossRef]
7. Glowacz, A.; Królczyk, G.; Antonino-Daviu, J.A. *Symmetry in Mechanical Engineering*; MDPI: Basel, Switzerland, 2020.
8. Modrea, A.; Munteanu, V.M; Pruncu, I. Using the Symmetries in the Civil Engineering. An overview. *Procedia Manuf.* **2020**, *46*, 906–913. [CrossRef]
9. Montoya, F.G.; Navarro, R.B. *Symmetry in Engineering Sciences*; MDPI: Basel, Switzerland, 2019.
10. Qui, W.; Yuan, J.; Ukwatta, E.; Sun, Y.; Rajchl, M.; Fenster, A. Prostate Segmentation: An Efficient Convex Optimization Approach With Axial Symmetry Using 3-D TRUS and MR Images. *IEEE Trans. Med. Imaging* **2014**, *33*, 947–960.
11. Wu, Y.; He, F.; Han, S. Collaborative CAD Synchronization Based on a Symmetric and Consistent Modeling Procedure. *Symmetry* **2017**, *9*, 59. [CrossRef]
12. Tyler, C.W. *Human Symmetry Perception and its Computational Analysis*; Psychology Press: Abingdon, UK , 1996.
13. Bertamini, M., Makin, A.D. Brain activity in response to visual symmetry. *Symmetry* **2014**, *6*, 975–996. [CrossRef]
14. Podgorelec, D.; Lukač, L.; Žalik, B. Reflection symmetry detection in Earth observation data. *Sensors* **2023**, *23*, 7426. [CrossRef]
15. Wang, Z.; Tang, Z.; Zhang, X. Reflection Symmetry Detection Using Locally Affine Invariant Edge Correspondence. *IEEE Trans. Image Process.* **2015**, *24*, 1297–1301. [CrossRef]
16. Mitra, N.J.; Pauly, M.; Wand, M.; Ceylan, D. Symmetry in 3D Geometry: Extraction and Applications. *Comput Graph Forum.* **2013**, *32*, 1–23. [CrossRef]
17. Abu-Faraj, M.; Al-Hyari, A.; Alqadi, Z. A Complex Matrix Private Key to Enhance the Security Level of Image Cryptography. *Symmetry* **2022**, *14*, 664. [CrossRef]
18. Chen, Y.; Zhan, S.; Xu, W.; Martin, R.R.; Cheng, Z.-Q. Parametric 3D modeling of a symmetric human body. *Comput Graphic.* **2019**, *81*, 52–60. [CrossRef]
19. Žalik, B.; Strnad, D.; Kohek, Š.; Kolingerová, I.; Nerat, A.; Lukač, N.; Podgorelec, D. A Hierarchical Universal Algorithm for Geometric Objects' Reflection Symmetry Detection. *Symmetry* **2022**, *14*, 1060. [CrossRef]
20. Petitjean, M. A definition of symmetry. *Symmetry Cult. Sci.* **2007**, *18*, 99–119.
21. Prantl, M.; Váša, L.; Kolingerová, I. Symmetry-aware Registration of Human Faces. In Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2019), Prague, Czech Republic, 25–19 February 2019; Cláudio, A.P., Bouatouch, K., Braz, J., Eds.; Science and Technology Publications: Setúbal, Portugal, 2019; pp. 185–192.
22. Hruda, L.; Kolingerová, I.; Lávička, M.; Mañak, M. Rotational symmetry detection in 3D using reflectional symmetry candidates and quaternion-based rotation parameterization. *Comput Aided Geom D.* **2022**, *98*, 102138. [CrossRef]
23. Zhang, D.; Lu, G. Review of shape representation and description techniques. *Pattern Recogn.* **2004**, *37*, 1–19. [CrossRef]
24. Van Otterloo, P.J. *A Contour Oriented Approach to Shape Analysis*; Pearson Education Limited: London, UK, 1991.
25. Sheynin, S.; Tuzikov, A.; Volgin, D. Computation of symmetry measures for polygonal shapes. In Proceedings of the 8th International Conference on Computer Analysis of Images and Patters (CAIP'99), Ljubljana, Slovenia, 1–3 September 1999; Lecture Notes in Computer Science 1689; Solina, F.; Leonardis, A., Eds.; Springer: Berlin, Germany, 1999; pp. 183–190.
26. Yang, X.; Adluru, N.; Latecki, L.; Bai, X.; Pizlo, Z. Symmetry of shapes via self-similarity. In Proceedings of the 4th International Symposium on Advances in Visual Computing (ISVC 2008), Las Vegas, NV, USA, 1–3 December 2008; Lecture Notes in Computer Science 5359; Bebis, G., Boyle, R., Parvin, B., Koracin, D., Remagnino, P., Porikli, F., Peters, J., Klosowski, J., Arns, L., Chun, Y.K., et al., Eds.; Springer: Berlin, Germany, 2008; pp. 561–570.
27. Kushnir, O.; Fedotova, S.; Seredin, O.; Karkishchenko, A. Reflection symmetry of shapes based on skeleton primitive chains. In Proceedings of the 5th International Conference on Analysis of Images, Social Networks and Text (AIST 2016), Yekaterinburg, Russia, 7–9 April 2016; Communications in Computer and Information Science 661; Ignatov, D.I., Khachay, M.Y., Labunets, V.G., Loukachevitch, N., Nikolenko, S.I., Panchenko, A., Savchenko, A.V., Vorontsov, K., Eds.; Springer: Cham, Switzerland, 2016; pp. 293–304.
28. Karkischenko, V.; Mnukhin, V. Fourfold symmetry detection in digital images based on finite Gaussian fields. In Proceedings of the First International Scientific Conference Intelligent Information Technologies for Industry (IITI'16), Sochi, Russia, 16–21 May 2016; Advances in Intelligent Systems and Computing 451; Abraham, A., Kovalev, S., Tarassov, V., Snášel, V., Eds.'; Springer: Cham, Switzerland, 2016; pp. 153–162.
29. Zahn, C.T.; Roskies, R.Z. Fourier descriptors for plane closed curves. *IEEE Trans. Comput.* **1972**, *C-21*, 269–281. [CrossRef]
30. Yip, R.; Tam, P.; Leung D. Application of elliptic Fourier descriptors to symmetry detection under parallel projection. *IEEE Trans. Pattern Anal.* **1994**, *16*, 277–286. [CrossRef]

31. Loy, G.; Eklundh, J.-O. Detecting Symmetry and Symmetric Constellations of Features. In Proceedings of the Computer Vision (ECCV 2006), Graz, Austria, 7–13 May 2006; Lecture Notes in Computer Science 3952; Leonardis, A., Bischof, H., Pinz, A., Eds.; Springer: Berlin, Germany, 2006; pp. 508–521.
32. Lowe, D.G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision* **2004**, *60*, 91–110. [CrossRef]
33. Duda, R.O.; Hart, P.E. Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Comm ACM.* **1972**, *15*, 11–15. [CrossRef]
34. O'Mara, D.; Owens, R. Measuring bilateral symmetry in digital images. In Proceedings of Digital Processing Applications (TENCON '96), Perth, Australia, 26–27 November 1996; IEEE: Piscataway, NJ, USA, 1996; pp. 151–156.
35. Sun, C.; Si, D. Fast reflectional symmetry detection using orientation histograms. *Real-Time Imaging* **1999**, *5*, 63–74. [CrossRef]
36. Mestetskiy, L.M.; Zhuravskaya, A. Mirror symmetry detection in digital images. In Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications–VISIGRAPP 2020 (4: VISAPP), Valletta, Malta, 27–29 February 2020; ; Farinella, G.M., Radeva, P., Braz, J., Eds.; Science and Technology Publications: Setúbal, Portugal, 2020 pp. 331–337.
37. Hauagge, D. C.; Snavely, N. Image matching using local symmetry features. In Conference on Computer Vision and Pattern Recognition (CVPR 2012), Providence, RI, USA, 16–21 June 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 206–213.
38. van Gool, L.; Moons, T.; Ungureanu, D.; Pauwels, E. Symmetry from shape and shape from symmetry. *Int. J. Robot Res.* **1995**, *14*, 407–424. [CrossRef]
39. Derrode, S.; Ghorbel, F. Shape analysis and symmetry detection in gray-level objects using the analytical Fourier–Mellin representation. *Signal Process.* **2004**, *84*, 25–39. [CrossRef]
40. Gnutti, A.; Guerrini, F.; Leonardi, D. Combining appearance and gradient information for image symmetry detection. *IEEE Trans. Image Process.* **2021**, *30*, 5708–5723. [CrossRef] [PubMed]
41. Zabrodsky, H.; Peleg, S.; Anvir, D. Hierarchical symmetry. In Proceedings of the 11th IAPR International Conference on Pattern Recognition, The Hague, The Netherlands, 30 August–1 September 1992; IEEE: Piscataway, NJ, USA, 1992; Volume III Conference C: Image, Speech and Signal Analysis, pp. 9–12.
42. Wang, Y.; Xu, K.; Li, J.; Zhang, H.; Shamir, A.; Liu, L.; Cheng, Z.; Xiong, Y. Symmetry hierarchy of man-made objects. *Comput Graph Forum.* **2011**, *30*, 287–296. [CrossRef]
43. Li, Y.; Mo, K.; Shao, L.; Sung, M.; Guibas, L. Learning 3d part assembly from a single image. In Proceedings of the Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020; Proceedings Part VI 16; Springer International Publishing: New York, NY, USA; pp. 664–682.
44. Paschalidou, D.; Gool, L.V.; Geiger, A. Learning unsupervised hierarchical part decomposition of 3d objects from a single rgb image. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 23–29 June 2020, Curran Associates, Inc.: Red Hook, NY, USA, 2020; pp. 1060–1070.
45. Xue, F.; Lu, W.; Chen, Z.; Webster, C.J. From LiDAR point cloud towards digital twin city: Clustering city objects based on Gestalt principles. *ISPRS J. Photogramm. Remote Sens.* **2020**, *167*, 418–431. [CrossRef]
46. Villanueva, A.J.; Marton, F.; Gobetti, E. Symmetry-aware Sparse Voxel DAGs (SSVDAGs) for compression-domain tracing of high-resolution geometric scene. *J. Comput. Graph. Tech. (JCGT)* **2017**, *6*, 1–30.
47. Madoš, B.; Chovancová, E.; Chovanec, M.; Ádám, N. CSVO: Clustered Sparse Voxel Octrees—A Hierarchical Data Structure for Geometry Representation of Voxelized 3D Scenes. *Symmetry* **2022**, *14*, 2114. [CrossRef]
48. Gonzalez, R.C.; Woods, R.E. *Digital Image Processing*, 4th ed.; Pearson: New York, NY, USA, 2018.
49. De Berg, M.; van Kreveld, M.; Overmars, M.; Schwarzkopf, O. *Computational Geometry: Algorithms and Applications*, 2nd ed.; Springer: Berlin, Germany, 2000.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.