Computational Aesthetics in Graphics, Visualization, and Imaging (2011)
D. Cunningham and T. Isenberg (Editors)

Symmetry Hybrids

B. Cullen[1] and C. O'Sullivan[1]

[1]Trinity College Dublin

Abstract

How we perceive the world is strongly governed by symmetry. Symmetry presents itself in both natural and man-made structures giving aesthetic appeal to the world. This paper presents an approach to form intuitive tree based representations that minimally describe input patterns. We explore how new hybrid patterns can be generated by grafting different symmetry trees together. A new algorithm is proposed to generate new hybrid patterns that maintain the overall appearance of the inputs while allowing control over the amount of variation generated.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

Symmetry is prevalent in both man-made and natural objects and plays a key role in aesthetics. Indeed, how we perceive the world is largely governed by symmetry [Koh92]. Patterns of repeating elements are present all around us from architecture to flower gardens.

Inverse procedural modelling involves detecting symmetry relations and representing them with a set of production rules. Procedural descriptions can accurately reproduce the input pattern while offering a highly compressed representation. The rules provide high level editing capabilities and can produce variation if desired.

Many patterns have a similar appearance despite containing different repeating elements. Building facades are composed of differing architectural elements while the symmetry relations between them are very similar. Cornices above windows have widely different appearance among architectural styles and yet we can identify them based on their symmetry relation with the windows.

This paper explores how symmetry can be recognised and used to generate hybrid patterns from examples. By interchanging similar symmetry relations between patterns we can produce many variations while preserving the overall look of the input patterns. We propose the following contributions:

1. A technique to detect hierarchical symmetries and represent them as trees.
2. A new algorithm to generate hybrid patterns by grafting symmetry trees together.

We review previous work in the area of symmetry detection and inverse procedural modelling in Section 2. We then discuss creating a tree based representation of the hierarchical symmetry relations in patterns (Section 3). Our approach to generate hybrid patterns using these symmetry trees is then described in Section 4. Finally, we demonstrate how this method can be used in a variety of applications (Section 5).

2. Related Work

Symmetry is an important property of the structure of many objects and plays an important role in human cognition [Ley92, Koh92]. Typical man-made structures exhibit symmetry and repeated elements that could be used to generate procedural descriptions.

Early approaches to symmetry detection were based on 2D data sets and attempted to reduce the problem to a 1D pattern matching problem [Ata85, WVW85]. Later, a variety of techniques were proposed to tackle the problem in 3D. Some of the proposed approaches are based on octree traversal [MIK93], Gaussian images [SS97], singular value decomposition [SS06] and generalised moments [MSHS06]. However, these methods can only detect perfect symmetries and fail to detect the imperfect symmetries present in noisy data.

DOI: 10.2312/COMPAESTH/COMPAESTH11/033-038

Recently a number of algorithms have been proposed to overcome this problem. Mitra et al. [MGP06] propose a technique to detect partial symmetries within a 3D model. They map the rotation and translation between pairs of points to transformation space, similar to the Hough transform. When all point pairs have been mapped, it is possible to use mean shift clustering [CM02] to find the relations of strongest symmetry. To complement this approach, Pauly et al. [PMW*08] present a framework to detect symmetries of regularly repeated elements within a structure. Their work has been extended to work with elements that are arranged in curved paths [YM09]. Bokelah et al. [BBW*09] report improved performance by considering feature lines to reduce the search space and can detect repeated elements that are not restricted to a regular pattern.

Interest has grown in using extracted symmetry information to create procedural rules to reproduce the input model with some variation and high level editing controls. The automatic generation of rules is far from a solved problem. The edge based subdivision algorithm presented by Müller et al. [MZWVG07] naturally leads to a derived rule set based on the split grammar proposed in [WWSR03]. However, their subdivision algorithm is restricted to grid based patterns, notably buildings, and cannot handle architecture styles that have large variation on different floors, thereby breaking the grid structure. Ijiri et al. [IMIM08] present an example based framework that can synthesise large patterns of elements given an input seed. The user inputs a small arrangement of elements and the system can automatically determine the connectedness and synthesise large patterns with different variations by applying noise. The user can also guide the synthesis directions. Most recently, Stava et al. [SBM*10] produced a system that can generate an L-System rule set given a vector image. They can produce L-Systems that model complex patterns and allow a high level of control over synthesised results

To date there has been little work on using symmetry information of different input examples to generate highly varying hybrid patterns. In this paper we address this challenge and demonstrate the usefulness of such a system.

3. Symmetry Detection

Symmetry plays an important role in the aesthetic appeal of artistic patterns. Detecting the underlying symmetry relations that describe such patterns has long been of interest in Computer Science. Most recently, Stava et al. [SBM*10] proposed a system capable of detecting the atomically repeating shapes present in vector patterns, and the symmetry relations between them. We extend their approach, representing the hierarchical structure of patterns with trees, which we will later use to derive new hybrid patterns.

We take as input a set of labelled shapes with known coordinates representing the atomic elements within an input pattern (e.g., segmented windows and doors from a photograph of a building). These can be segmented from images using computer vision algorithms such as the GrabCut technique proposed by Rother et al. [RKB04]. However, appropriate segmentation techniques vary largely between different image patterns, leaving a detailed discussion beyond the scope of this paper.

Following from Stava et al. [SBM*10], we define transformations between pairs of identical shapes a and b as $t_{ab} = t_a^{-1} t_b$ where t_a is a 3-tuple $\langle \alpha, d, \beta \rangle$ representing the position in polar coordinates (angle α and distance d), and orientation β of shape a (see Figure 1a). As in Mitra et al. [MGP06] we take the transformations between every pair of shapes and perform mean shift clustering within this transformation space. Once this set of clusters has been determined, it is possible to find symmetries of regularly repeating elements.

Patterns are generally composed of many regularly repeating elements, forming hierarchical structures. Reliably detecting hierarchical symmetries is thus the foundation of our approach and warrants detailed discussion. Consider the pattern in Figure 1b which has a noticeable hierarchical structure consisting of cyclic symmetry groups. The higher level symmetry composing the large circle does not present itself in transformation space until we determine the symmetry grouping of the smaller circles.

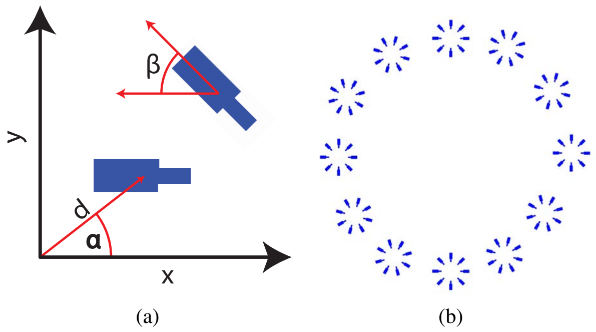

Figure 1: (a) Shapes are represented with polar coordinates (α, d) and orientation β. (b) Patterns can be composed of hierarchical symmetry relations between shapes. The blue shapes form cyclic groups of 45° which in turn form a cyclic group of 30°

Hierarchical structures like the one above produce many dense clusters of unimportant symmetries in transformation space (i.e., they do not minimally describe the structure). This problem can be addressed by grouping transformation clusters together that are powers of each other into weighted sets. This is illustrated in Figure 2 where the transformation t_2 is a power of t_1:

$$t_2 = t_1 t_1 t_1 = t_1^3$$

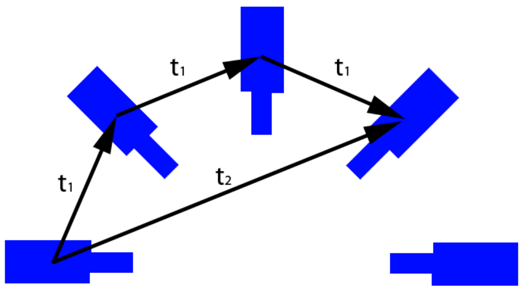

Figure 2: Larger transformations are composed of chains of smaller transformations within the same symmetry group. Here, transformation t_2 is equivalent to performing t_1 three times.

Transformations of lesser magnitude generally represent more important symmetry relations than those of higher magnitude. In order to exploit this law of proximity, weights are assigned to transformation clusters according to their magnitude, $w(t_i) = \sqrt{\alpha^2 + d^2 + \beta^2}$. Let T be the list of all transformation cluster centres, sorted by their assigned weights.

$$T = \langle t_1, t_2, t_3, \ldots, t_n \rangle | w(t_i) \leq w(t_{i+1})$$

A new set $S_i \subseteq T$ is created containing all transformations t_k which are some power of t_1 (the symmetry with lowest magnitude). To account for variance in the data we define two transformations t_k and t_1^m to be equivalent if the distance between them is less than some small threshold ε:

$$t_k \in S_i | d(t_k, t_1^m) < \varepsilon$$

These cluster centres are then removed from the list so that $T = T \backslash S_i$ and the process repeated until T is empty. We are left with a collection of sets S which correspond to possible symmetry relations among the input shapes. We take the set S_i with highest cardinality and lowest weight and perform the energy minimisation technique of Pauly et al. [PMW*08] on its members to find the symmetry relation that minimises error.

Shapes are then grouped into patches according to this symmetry relation. This whole process is repeated by constructing a new transformation space consisting of transformations between patches. Each iteration of this process creates one node, thereby forming a tree of symmetry relations.

3.1. Grouping

Patterns are usually composed of more than one type of shape. It is therefore necessary to find relations between different types of shapes in order to fully describe the pattern. Shapes are grouped based on the following precedence rules presented in [WXL*11]:

1. Identical shapes are grouped by symmetry before grouping with other shapes.
2. Non-identical shapes are grouped first if they possess equivalent grouping symmetries.

Figure 3 illustrates these two scenarios.

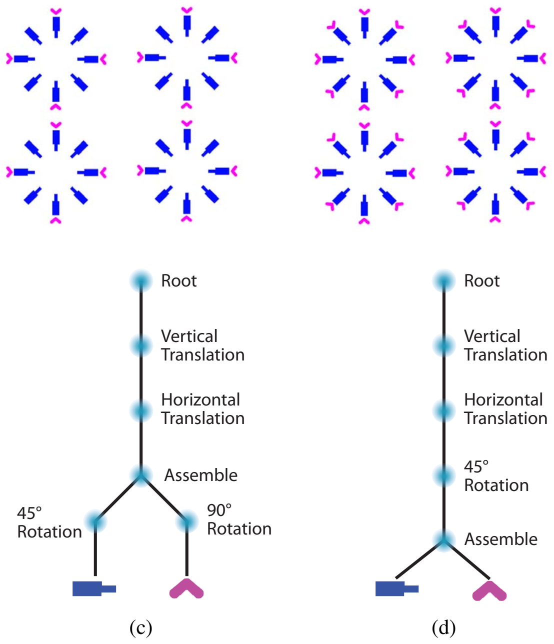

Figure 3: (a) Shapes must be grouped into their individual symmetry groups before being grouped together. (b) Shapes must be grouped together before being grouped into the same symmetry group.

4. Generating Hybrids

Having determined the symmetry trees for a set of input patterns, it is then possible to create a hybrid symmetry tree. This hybrid tree will be used to generate a production system capable of synthesising new varying patterns. Hybrid patterns should exhibit the symmetry relations of all input examples. We accomplish this by finding matching subtrees between the input trees and connecting their nodes together. We use the following matching criteria to match subtrees:

1. Nodes that represent rotational symmetry are matched with other rotational symmetry nodes regardless of their angle of rotation.
2. Nodes that represent translational symmetry can only be matched with nodes that represent collinear translations.
3. Terminal and assembly nodes match each other.

By specifying the minimum depth that subtrees must match, the number of matching nodes is constrained, thereby controlling the amount a hybrid pattern varies from the input examples (See Figure 5). The algorithm to accomplish this is given in Figure 4, while the effect of changing the matching depth is demonstrated in Figure 6.

```
hybridizeTrees(tree,searchTree,minDepth)
  foreach subtree in searchTree
    [matchTree score] = find(subtree,tree);
    if(score > minDepth)
      matchTree.addSibling(subtree);
      subbTree.addSibling(matchTree);
    end
  end
end
```

Figure 4: Algorithm to control the amount of variation a hybrid can exhibit. *Find* searches for a subtree within a bigger *tree*, returning all nodes in *tree* where the *subtree* matches and the matching score (number of matching nodes).

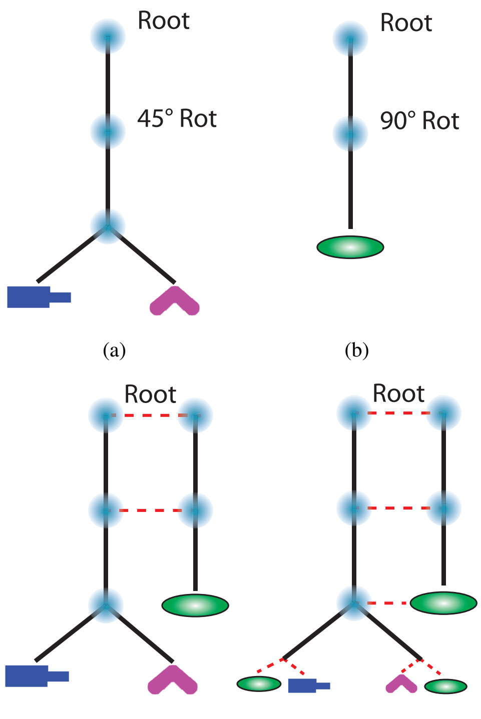

Figure 5: (a–b) Graphs before hybridisation. (c) Subtree matching with matching depth of 2. (d) Subtree matching with matching depth of 1. Broken red line indicates where nodes have been connected as siblings.

4.1. Production System

The detected symmetry tree is used to produce a procedural description of the pattern. The rules are specified using a script with a parameterised L-System [PL90] style syntax:

$$Predecessor \rightsquigarrow Command(params)\{Successor\}:\ Probability$$

Command is carried out on the shape with ID *Predecessor* and the resulting output shapes are given the ID *Successor*. Multiple rules can be specified for the same *Predecessor* and one is chosen at random based on its *Probability* value. This allows variability among generated patterns.

A rule is created for each node in the symmetry tree. *Predecessor* is given the node name while the *Command* and *Successor* symbols are determined from the symmetry relation the node represents and its children. Several rules may have the same *Predecessor* corresponding to sibling nodes in the hybrid symmetry tree. Only two commands are required to reproduce all input patterns:

Assemble(α, d, β)
: Assembles two successor shapes together with a transformation between them.

Symmetry(α, d, β, n)
: Repeats the transformation n times, creating n successor symbols. If $n < 1$ the transformation is repeated until there is no space left on the canvas.

These commands are based on shape grammar commands as described in [WWSR03]. However, unlike shape grammars this production system can produce overlapping symbols and is more suitable for generating patterns, while offering more control over the number of repetitions than L-Systems.

5. Results

Generating hybrids from symmetry has a number of useful applications ranging from automatic content generation for virtual worlds to artistic designs for wallpaper or ornaments. We have tested this technique within a number of domains. Figure 7 demonstrates hybrid patterns from various input examples. In all cases only two input patterns were used to generate the varying content illustrated. Only a few input examples are required to produce hundreds of variations. It is even possible to produce fractal patterns by matching symmetry relations at different hierarchy levels between patterns. A few photographs of buildings are sufficient to produce many new architectural designs, complementing procedural city generation.

6. Conclusion

We have presented a technique to create tree based representations of the hierarchical symmetries present in patterns. In Section 4 we proposed a new algorithm to generate hybrid patterns from detected symmetry hierarchies. Our results show that many new patterns can be generated from only a few input examples. Generation of hybrid patterns has widespread artistic applications in computer graphics, complementing existing procedural modelling applications such as procedural building generation [CO11] and vector pattern synthesis [SBM*10], to name a few. However, our approach

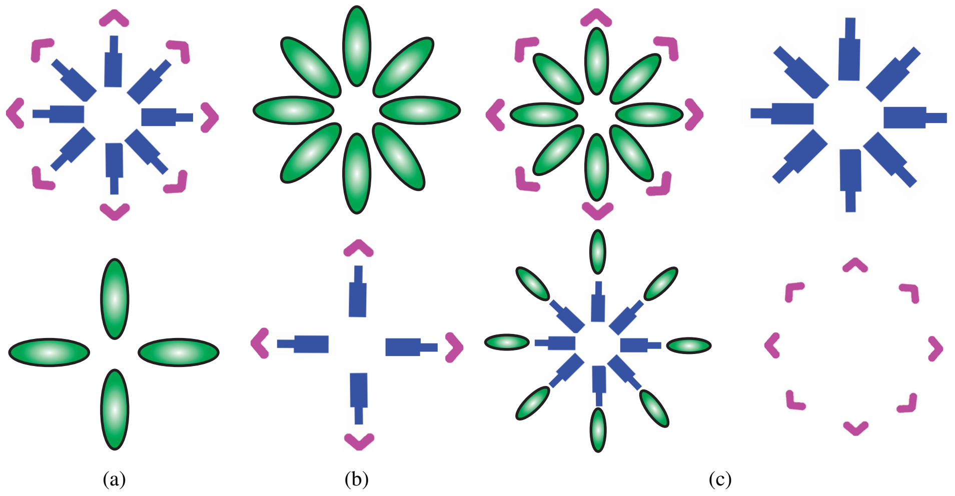

Figure 6: (a) Input patterns. (b) Subtree matching depth set to 2. Some variation is possible while preserving the input structure. (c) Subtree matching depth set to 1. Much more variation is possible as less of the original structures are preserved.

is limited in that it offers very little control over the outputs. This works well for abstract patterns but in some applications such as building facade generation, certain architectural elements may not fit well together and finer control is needed. Future work would involve a user-trained system that can help prune unwanted hybridisations.

References

[Ata85] Atallah M. J.: On symmetry detection. *IEEE Trans. Comput. 34*, 7 (1985), 663–666. 1

[BBW*09] Bokeloh M., Berner A., Wand M., Seidel H., Schilling A.: Symmetry detection using line features. *Computer Graphics Forum (Proceedings of Eurographics)* (2009). 2

[CM02] Comaniciu D., Meer P.: Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell. 24*, 5 (2002), 603–619. 2

[CO11] Cullen B., O'Sullivan C.: A caching approach to real-time procedural generation of cities from GIS data. *Journal of WSCG 19*, 1–3 (2011), 119–126. 4

[IMIM08] Ijiri T., Mêch R., Igarashi T., Miller G.: An example-based procedural system for element arrangement. *Computer Graphics Forum 27*, 2 (2008), 429–436. 2

[Koh92] Kohler W.: *Gestalt Psychology: An Introduction to New Concepts in Modern Psychology*. Liveright Publishing Corporation, July 1992. 1

[Ley92] Leyton M.: *Symmetry, Causality, Mind*. MIT Press, Apr. 1992. 1

[MGP06] Mitra N. J., Guibas L. J., Pauly M.: Partial and approximate symmetry detection for 3D geometry. *ACM Trans. Graph. 25*, 3 (2006), 560–568. 2

[MIK93] Minovic P., Ishikawa S., Kato K.: Symmetry identification of a 3-D object represented by octree. *IEEE Trans. Pattern Anal. Mach. Intell. 15*, 5 (1993), 507–514. 1

[MSHS06] Martinet A., Soler C., Holzschuch N., Sillion F. X.: Accurate detection of symmetries in 3D shapes. *ACM Trans. Graph. 25*, 2 (2006), 439–464. 1

[MZWVG07] Müller P., Zeng G., Wonka P., Van Gool L.: Image-based procedural modeling of facades. *ACM Trans. Graph. 26* (July 2007). 2

[PL90] Prusinkiewicz P., Lindenmayer A.: *The algorithmic beauty of plants*. Springer-Verlag New York, Inc., 1990. 4

[PMW*08] Pauly M., Mitra N. J., Wallner J., Pottmann H., Guibas L. J.: Discovering structural regularity in 3d geometry. *ACM Trans. Graph. 27*, 3 (2008), 1–11. 2, 3

[RKB04] Rother C., Kolmogorov V., Blake A.: "GrabCut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph. 23*, 3 (2004), 309–314. 2

[SBM*10] Stava O., Benes B., Mech R., Aliaga D., Kristof P.: Inverse procedural modeling by automatic generation of l-systems. *Eurographics (Conditionaly accepeted)* (2010). 2, 4

[SS97] Sun C., Sherrah J.: 3D symmetry detection using the extended Gaussian image. *IEEE Trans. Pattern Anal. Mach. Intell. 19*, 2 (1997), 164–168. 1

[SS06] Shah M. I., Sorensen D. C.: A symmetry preserving singular value decomposition. *SIAM J. Matrix Anal. Appl. 28*, 3 (2006), 749–769. 1

[WVW85] Woo T., Volz R., Wolter J.: Optimal algorithms for symmetry detection in two and three dimensions. *The Visual Computer 1*, 1 (July 1985), 37–48. 1

[WWSR03] Wonka P., Wimmer M., Sillion F., Ribarsky W.: Instant architecture. *ACM Trans. Graph. 22*, 3 (2003), 669–677. 2, 4

[WXL*11] Wang Y., Xu K., Li J., Zhang H., Shamir A., Liu L., Cheng Z., Xiong Y.: Symmetry hierarchy of man-made objects. *Computer Graphics Forum 30*, 2 (2011), 287–296. 3

[YM09] Yeh Y., Měch R.: Detecting symmetries and curvilinear arrangements in vector art. *Computer Graphics Forum 28*, 2 (2009), 707–716. 2

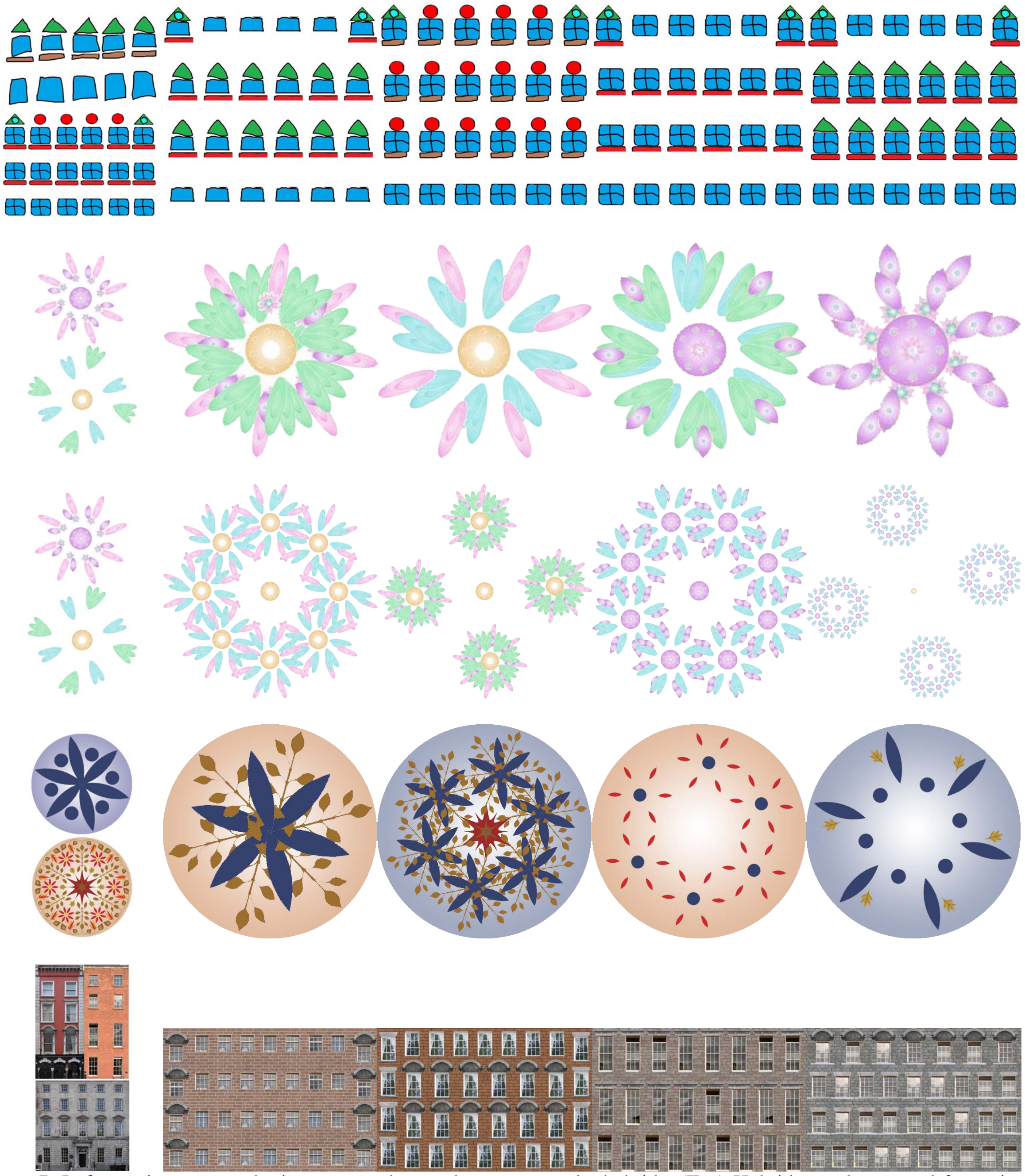

Figure 7: Leftmost images are the input examples used to generate the hybrids. (Top) Hybrids can be created from simple sketches. (Second Row) Abstract floral art. Hybrids may assist artists in coming up with new designs. (Third Row) The system can produce fractal shapes by observing similar symmetries among different hierarchy levels. (Fourth Row) Ornamental designs. (Bottom Row) Procedural building generation. Only a few photographs are necessary to produce hundreds of new buildings.