Symmetry Detection Competition
Summary and Results of Round 2

Executive Summary

We completed the second round of the symmetry detection competition and presented results at a dedicated workshop hosted at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2011 in Colorado Springs, Colorado.

The competition was divided into three parts, each focusing on one of three types of symmetries: reflection, rotation and translation respectively. For each symmetry category, we collected images depicting objects with representative symmetry features. In order to minimize bias towards specific symmetries, we also obtained a large variety of symmetry images from professional and amateur photographers who signed up and submitted images to our Flickr photosharing website (http://www.flickr.com/groups/symmetrycompetition/). We collected a total of four training and six test sets, totaling 124 images.

Each image was annotated, yielding a total of 167 reflection and rotation symmetries and over two thousand wallpaper tiles for translation symmetry. During the annotation phase, we identified and specified several ambiguities that can arise during the labeling process. In all cases of ambiguities a trade off between local and global context had to be made before it could be resolved. Because this trade off is in almost all cases subjective in nature, we have marked such symmetries as ambiguous and discounted any detections by the tested algorithms.

We received eight submissions for symmetry detection, three for reflection, two for rotation and three for translation symmetry. Adding one baseline algorithm to each symmetry group for comparison, we evaluated a total of eleven algorithms.

The evaluation process was completely automated, counting the number of correct detections (true positives) and the number of incorrect detections (false positives). Detection performance is then reported in terms of precision and recall rates., for which the later only considers the number of true positives vs the total number of symmetries present, while the former also considers and penalizes for the number of false positives. The quite remarkable outcome of all evaluations is that overall, none of the submitted algorithms performed better than their respective baseline algorithm (including execution speed of the algorithms). In comparison, tested algorithms performed best on reflection symmetry, second best on rotation symmetry and last translation symmetry.

However, looking at sub categories of reflection symmetries, we found that one algorithm performed superior to all other tested algorithms on real images with multiple reflection axis present (judged on recall rate). A different algorithm outperforms all other algorithms on real and synthetic images depicting only a single reflection symmetry, when the number false positives has been taken into account (judged on precision rate).

Overall, our efforts have been successful. We established a testbed for the evaluation of symmetry detection algorithms, devised evaluation metrics and automated the evaluation process. We tested our process on eleven algorithms and established a performance baseline that can be used as reference for future work on symmetry detection.
Method
In this section we provide background information on our data collection and annotation methods and inform the reader on how the execution and evaluation of submitted algorithms has been carried out.

Data Sets and Groundtruth Annotation

Data Collection
The competition was divided into three parts, each focusing on one of three types of symmetries: reflection, rotation and translation respectively. For each symmetry category, we collected images depicting objects with representative symmetry features from personal photos sets of our students. In order to minimize bias towards specific symmetries, we also obtained a large variety of symmetry images from professional and amateur photographers who signed up and submitted images to our Flickr photosharing website (http://www.flickr.com/groups/symmetrycompetition/). We collected a total of four training and six test sets, totaling 124 images. Figure 1 illustrates four of the six test image sets.

Figure 1: Four of the six image data sets are shown. (Top Left) Reflection Symmetry, (Top Right) Rotation Symmetry, (Bottom Right) Translation Symmetry (Urban Buildings), (Bottom Left) Translation Symmetry
Categorization of Image Data Sets

For each symmetry type we split the obtained datasets into a number of relevant sub categories. For example, we split all 30 images of the reflection symmetry test set into four groups: images depicting synthetic vs real images and images depicting a single vs. multiple reflection symmetries. The purpose of this categorization is to evaluate and identify strength or weaknesses of the tested algorithms on specific image categories. The individual number of images and symmetries in each category can be extracted from Table 1.

For rotation symmetry we replaced the categories real and synthetic with discrete, continuous and deformed symmetry elements. The purpose here is to test for performance differences amongst images with discrete symmetry elements such as the pedals of a flower or the spikes of a car wheel, versus images with no discernable symmetry elements, such as a smooth ring. We also test images with deformed symmetry objects, in which the object is partially occluded or affected by some affine or perspective distortion. We expect algorithms to perform best on images with discrete symmetries and worst on images with deformed symmetry objects. Examples and the number of images and symmetries in each category are given in Table 2.

For translation symmetry we split the dataset into three categories: easy, medium and hard, each of which holds various challenges to the algorithm. The easy category holds images with clearly visible wallpaper structure and only mild affine distortions. The medium category holds images with strong wallpaper structure but more severe affine and perspective distortions, as well as irregular structures (e.g. wrinkles on clothing). The hard category is most challenging due to the presence of strong and distracting background clutter, which in some cases is unstructured and in some cases is itself structured in form of translational symmetry. Examples and the number of images and wallpaper texels in each category are given in Table 3.

Table 1: The dataset for reflection symmetry is divided into four categories, synthetic vs. real images, and images with single vs. multiple reflection axis. We use a total of 30 images with a combined count of 66 reflection axis.

<table>
<thead>
<tr>
<th>Category</th>
<th>Single</th>
<th>Multiple</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#Imgs</td>
<td>#Syms</td>
<td>#Imgs</td>
</tr>
<tr>
<td>Synthetic</td>
<td>8</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>38</td>
<td>30</td>
</tr>
<tr>
<td>Real</td>
<td>6</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>28</td>
<td>30</td>
</tr>
<tr>
<td>Total</td>
<td>14</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>66</td>
<td></td>
</tr>
</tbody>
</table>
Table 2: The dataset for rotation symmetry is divided into six categories, images with discrete vs. continuous symmetry elements, deformed symmetry objects, and images with single vs. multiple reflection axis. We use a total of 40 images with a combined count of 81 rotation symmetries.

<table>
<thead>
<tr>
<th>Category</th>
<th>Single</th>
<th>Multiple</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#Imgs</td>
<td>#Syms</td>
<td>#Imgs</td>
</tr>
<tr>
<td>Discrete</td>
<td>11</td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>Continuous</td>
<td>10</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>Deformed</td>
<td>7</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>Total</td>
<td>28</td>
<td>28</td>
<td>12</td>
</tr>
</tbody>
</table>

Table 3: The dataset for translation symmetry is divided into three categories, easy, medium and hard. We use a total of 31 images with a combined count over 2000 wallpaper tiles.

<table>
<thead>
<tr>
<th>Category</th>
<th>Easy</th>
<th>Medium</th>
<th>Hard</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#Images</td>
<td>#Texels</td>
<td>#Images</td>
<td>#Texels</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>536</td>
<td>10</td>
<td>1330</td>
</tr>
</tbody>
</table>

---

This table represents the dataset for rotation and translation symmetries, categorizing images into discrete, continuous, and deformed types, along with single or multiple symmetry representations. The dataset includes 40 images with 81 rotation symmetries and 31 images with over 2000 wallpaper tiles for translation symmetry.
Groundtruth Annotation

Employing the manual power of 30 students from our course on “Symmetry for Image Processing” at Penn State, each image was annotated using annotation programs, developed specifically for this purpose, and yielding a total of 167 reflection and rotation symmetries and over two thousand wallpaper tiles for translation symmetry. An example of annotation labels for each of the three symmetry groups is shown in Figure 2.

![Image with annotation labels](image)

Figure 2: Images with annotation labels for (Left) reflection, (Mid) rotation and (Right) translation symmetry.

Reflection symmetry axis are marked as a line with a start point $p_1=(x_1,y_1)$ and an end point $p_2=(x_2,y_2)$. The length of the line covers the respective support region of the annotated symmetry. For rotation symmetry an ellipse is defined that covers the maximal support region, with center point $c=(cx, cy)$, major and minor axis length $L=(a, b)$ and the orientation $\theta$ of major axis with respect to the image x-axis. For translation symmetry, a lattice is defined with a start point $P=(x, y)$ and a vector field $T$ (two vectors for each tile), and each tile represents the texel in a wallpaper pattern.

Annotation Ambiguities

During the annotation phase we identified a number of ambiguities that can arise during the labeling process. In all cases of ambiguities a trade off between local and global context seems to play a major role in deciding how the ambiguity can be resolved. Here, we give two examples from reflection symmetry:

1. Hierarchical Ambiguity
2. Shape Ambiguity

These highlight ambiguities caused by scale of context and object deformations. Looking at hierarchical ambiguity, we refer to Figure 3. Symmetry is defined as a transformation $g$ of a set of points $S$ such that $g(S)=S$. Traditionally $S$ represents the entire set of points, or in the case of a 2D space, the entire image. Given such a global definition of symmetry, only few true symmetries can be defined. However, to the human eye many more symmetries appear when viewed on a local rather than global scale. When looking at Shape Ambiguity we are confronted with the problem that the definition of symmetry $g(S)=S$ seldom holds true in practice. In real images, symmetric elements rarely are exact copies of each other. Instead, slight deformation of shape and subtle differences in texture, color or lighting are common place, yet to the human eye such differences are often of little significance when judging symmetry (see Figure 4). Similar scenarios of ambiguity can be constructed for rotation and translation symmetry as well. Because a trade off in weighting local versus global symmetry elements is required in almost all cases and is extremely subjective in nature, we mark such symmetries as...
ambiguous and discount any detection by the tested algorithms so that neither the number of true positives nor the number of false positives is affected.

Figure 3: Hierarchical Ambiguity of reflection symmetry. (Top Left) Without local context, symmetry is defined over the entire image. (Top Right) When using a subset of the 2D plane (local context) many different reflection symmetries can be defined. (Bottom) An example of scale dependent annotation of reflection symmetry (blue lines).

Figure 4: Shape Ambiguity in reflection symmetry. (Top Row) Two squares form a perfect reflection symmetry along their mirror axis. However, as one of the squares changes into a triangle, the boundary between valid and invalid symmetry fades. (Bottom row) Real world examples of shape ambiguity. While reflection symmetry within an object seems legitimate, symmetry between objects seems to be more subjective and application dependent.
Eventually, what is required is to define a symmetry transformation that is invariant to small and local disturbances of object shape and appearance. A more formal definition of such symmetry ambiguities is required and we see this as a very interesting and essential future work.

**Contestants and Algorithm Execution**

We received eight submissions for symmetry detection, three for reflection, two for rotation and three for translation symmetry. Adding one baseline algorithm to each symmetry group for comparison, we evaluated a total of eleven algorithms. The following table shows all contestants and baseline algorithms used in this competition:

<table>
<thead>
<tr>
<th>Symmetry Group</th>
<th>Contestant(s)</th>
<th>Institution(s)</th>
<th>Algorithm/Code</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reflection</strong></td>
<td>Mo and Draper</td>
<td>Colorado State, USA</td>
<td>Matlab, SPE</td>
</tr>
<tr>
<td></td>
<td>Eckhardt</td>
<td>Frauenhofer ISOB, Germany</td>
<td>Matlab, Pipelines</td>
</tr>
<tr>
<td></td>
<td>Kondra and Petrosino</td>
<td>Uniparthenope, Napoli, Italy</td>
<td>Matlab, Mex-Files, SPE</td>
</tr>
<tr>
<td></td>
<td>Loy and Eklundh</td>
<td>Windows Executable, (publicly available)</td>
<td></td>
</tr>
<tr>
<td><strong>Baseline</strong></td>
<td>Kim, Cho and Lee</td>
<td>Seoul National University, South Korea</td>
<td>Matlab</td>
</tr>
<tr>
<td></td>
<td>Kondra and Petrosino</td>
<td>Uniparthenope, Napoli, Italy</td>
<td>Matlab</td>
</tr>
<tr>
<td></td>
<td>Loy and Eklundh</td>
<td>Windows Executable, (publicly available)</td>
<td></td>
</tr>
<tr>
<td><strong>Rotation</strong></td>
<td>Cai</td>
<td>Polytechnic, Hongkong, China</td>
<td>Matlab, Mex-Files, GUI</td>
</tr>
<tr>
<td></td>
<td>Wu</td>
<td>University of North Carolina, USA</td>
<td>Windows Executable</td>
</tr>
<tr>
<td></td>
<td>Eckhardt</td>
<td>Frauenhofer ISOB, Germany</td>
<td>Matlab, Pipelines</td>
</tr>
<tr>
<td></td>
<td>Park, Brocklehurst, Collins and Liu</td>
<td>Pennsylvania State University, USA</td>
<td>Windows Executable, (publicly available)</td>
</tr>
</tbody>
</table>

Prior to the submission deadline, we provided potential contestants with a set of training images for each symmetry group. Contestants used these images to adapt and potentially fine-tune their free parameters in order to achieve best possible results on these training sets.

However, we nonetheless had various issues in executing the submitted algorithms on our image test set. While some algorithms came with a nice Graphical User Interface (GUI), it forced us to manually run the detection algorithm for each test image separately. Other

---

algorithms came in form of Matlab code and a single point of entry (SEP), which made automatic batch processing of all images easy. We also received algorithms that required a number of manual steps for pre-processing, and for which documentation was rather scarce. After communicating with the authors most of the issues could be resolved. Yet, several algorithms still had abnormal terminations on a subset of images.

**Algorithm Evaluation**

For all three symmetry groups the algorithm performance is measured in terms of precision and recall rates, which are two popular metrics for evaluating the correctness of a pattern recognition algorithm. Both can be seen as extended versions of sensitivity, a simple metric that computes the fraction of instances for which the correct result is returned.

When using precision and recall, the set of possible labels for a given detection is divided into two subsets, one of which is considered "true" or “correct” for the purposes of the metric, the other “false” or “incorrect”. Recall is then computed as the fraction of correct detections (true positives) among all instances that actually belong to the correct subset (number of groundtruth symmetries), while precision is the fraction of correct detections among those that the algorithm believes to belong to the relevant subset, which includes all those detections that are “true” (true positives) and “wrong” (false positives).

Precision = \( \frac{TP}{TP + FP} \)
Recall = \( \frac{TP}{TP + FN} \)

Precision can be seen as a measure of exactness or fidelity, whereas recall is a measure of completeness. In practice, a high recall means most symmetries have been correctly detected, without considering a possibly high number of incorrect detections (which would imply low precision). High precision means that most detections are a correct detections, although not all symmetries might have been found (which would imply low recall).

While for reflection and rotation symmetry evaluation, recall and precision rates are computed for the total number of true positives and false positives over all images, the recall and precision scores for translation symmetry is calculated for each image separately and then combined and averaged over the number of all images. Because the number of texels on which we count the number of correct detections varies vastly between images (e.g. some images have 200+ texels while others might have only twelve), this averaging is necessary to avoid any bias towards images with a high count of texels.

We now explain the process of determining the number of correct and incorrect detections for each symmetry group, which has been completely automated for this study.
Reflection Symmetry

For each detection result \( R \), we measure the angle between the detected symmetry axis (R) and the ground-truth axis (GT). We also measure the distance between the centers of both lines. A correct detection (true positive) is achieved, if the orientation between the two axis is less then some threshold \( t_1 \), and the distance between the two axis is less then some threshold \( t_2 \).

For \( t_1 \) and \( t_2 \) we choose \( t_1 = 10 \) degrees and \( t_2 = 20\% \) of the ground-truth axis length. Multiple valid detections (\( R_1, R_2 \)) can be clustered if they associate with the same ground-truth axis. One detection result, however, can only be associated with one ground truth axis. The size of the symmetry support region is not further considered.

Rotation Symmetry

For each detection result we measure the distance \( d \) between detected (C) and ground-truth symmetry center (\( C_{GT} \)). We also record the radius \( R \) of the detected symmetry region (and use the length of the major axis, if the detection algorithm reports an ellipse instead of a circle as support region). A correct detection (true positive) is achieved, if the distance \( d \) between detected and some ground-truth symmetry center is below some threshold \( t_1 \), and if the radius \( R \) of the detected support region is within some bounds \([b_1, b_2] \). For \( t_1 \) we choose \( t_1 = \text{max}(5, 0.02R_{GT}) \) and for \([b_1, b_2] \) we choose \( b_1 = 0.8* R_{GT} \) and \( b_2 = 1.2* R_{GT} \).

As with reflection symmetry, one detection result can match to only one ground-truth symmetry, but multiple different detections can be matched to one ground-truth center. Not considered in this evaluation are more detailed support region descriptors (such as an ellipse instead of a circle), the number of symmetry folds and the the type of symmetry (e.g. discrete or continuous).
Translation Symmetry

We count the number of correctly detected tiles in a lattice structure that encodes the wallpaper pattern present in an image. A quadrilateral lattice tile is correct if all its four corners match up to corners in the ground-truth lattice. Because many different lattice structures can correctly define a wallpaper pattern, a global matching between detected and recoded ground-truth lattice has to be devised first. We have created an automated method of lattice evaluation (Park et al., (2009)) that establishes a mapping between a detected lattice $T$ and the ground truth lattice $G$ by minimizing a distance cost-function between paired lattice points using a globally unique affine transformation to all detected lattice points. This method works very reliably in most cases. In rare cases, it can generate false positives due to odd boundary conditions caused by occluding objects or the edge of the image. For this reason, a visual inspection has been carried out to ensure that only texels that are occluded by half or less are counted as valid.

Results

We no present our results on the symmetry detection competition. For each symmetry group we show precision and recall rates overall and for each contestant and for each image category. Sample detection outputs and side-by-side comparisons can be found in the figures at the end of this document.

Reflection Symmetry

Figure 5: A global offset between ground truth (red) and detected lattice (dotted black) is found by minimizing the distance between all lattice points under some optimal global affine transformation, applied to all tiles simultaneously.

---

Overall, the algorithm by Loy and Eklundh (our baseline) outperforms all contestants with a recall rate of 0.68, compared to the algorithm by Kondra with a recall rate of 0.64 and that of Mo with a recall rate of 0.47. This performance advantage is due to a single image category, namely synthetic images, in which Loy and Eklundh outperform all others with a clearer margin (0.68 versus 0.55 and 0.53). In all other categories the algorithms by our contestants perform equally well or better than our baseline algorithm. Among the contestants, Kondra’s algorithm performs better than Mo’s algorithm (0.64 versus 0.47). In fact Kondra’s algorithm performs better than Mo’s in three out of four image categories. Multiple symmetry Axis: 0.65 versus 0.38, Synthetic Images: 0.55 versus 0.53, and Real Images: 0.75 versus 0.46. Only in the single symmetry axis category does Mo’s algorithm perform better than Kondra’s algorithm (0.93 versus 0.57).

When we look at precision rates, the picture changes. Kondra’s algorithm exhibits a relatively high number of false positives, when compared to Mo’s or Loy and Eklundh’s algorithm and therefore offers consistently less precision. Again, as with recall, Mo’s algorithm achieves best precision on images with a single reflection axis. In all other categories our baseline algorithm performs best.

We also compare execution times of all algorithms. Our baseline algorithm again performs best with an average execution time per image of 0.97 seconds. Mo’s algorithm compares favorably with 3.8 seconds, while Kondra’s algorithm takes on average 15.6 seconds per image. All computations were performed on a Windows Vista 64bit machine with an i7, 2.67G cpu (8 core), 6GB ram and used Matlab R2008b.
Overall, our baseline algorithm by Loy and Eklundh clearly outperforms all other algorithms by a significant margin: Recall=0.51 versus 0.21 (Kondra) and 0.16 (Kim). It also performs best in each image category. Amongst the contestants, Kondra’s algorithm performs overall better than Kim’s algorithm (Recall=0.21 versus 0.16). Kondra’s algorithm is especially better than Kim’s algorithm on images with single (0.31 versus 0.14) and deformed rotation symmetries (0.21 versus 0.12). On images with continuous symmetries, we observe that Kim’s algorithm has not produced a single correct detection. However, Kim’s algorithm performs better then Kondra’s algorithm on images with discrete (0.41 versus 0.626) and multiple rotation symmetries (0.17 versus 0.15). Again similar to reflection symmetry, when we look at precision rather than recall rates, our baseline algorithm performs best overall, while Kondra’s relatively high number of false positives changes the performance picture in favor of Kim’s algorithm in all image categories.
We also compare execution times of all algorithms. Our baseline algorithm again performs best with an average execution time per image of 0.59 seconds. Kondra’s algorithm now compares favorably with 6.9 seconds per image, while Kim’s algorithm takes on average 68.2 seconds per image.

**Translation Symmetry**

Evaluation of translation symmetry detection was unfortunately not as straight forward as it has been for reflection and rotation symmetry detection. The three submissions we have received did not end up conforming to the standard (see Figure 8) as laid out and published by us on our website.

Cai’s output is not a valid lattice structure

Wu’s output often only shows vanishing lines

![Figure 8: Issues with algorithm outputs for translation symmetry detection.](image)

**Issues with submitted algorithms**

The algorithm by Yunliang Cai requires user input to provide an initial guess for the lattice structure, making a fair comparison to other algorithms, in particular our baseline, impossible. Further complicating the evaluation is that Cai’s algorithm output format does not comply to our standard lattice format. We were unable to find a suitable
technique for converting their output format to ours. A direct communication with the authors would have been helpful, and was attempted but failed. The algorithm by Changchang Wu is primarily designed for 1D frieze pattern detection on building facades, and requires strong horizontal features for vanishing point detection. It nonetheless worked on some of our test images but failed on many others. The algorithm by Michaele Eckhardt was initially designed only for reflection symmetry detection but an attempt has been made by its author to modified it for translation symmetry detection as well. However, after a long series of back and forth communication, and the realization that the modification did not bring about the desired results, the algorithm was eventually withdrawn from the translation symmetry detection competition.

**Evaluation 1**

We attempted quantitative evaluation by transforming the output lattice by Cai’s algorithm into a valid lattice format when ever possible. We dismissed all results for which this transformation was not possible. The total number of test images for which all contestant reported valid results where then only four images (two from easy set, and one from each of medium and hard sets).

![Recall and Precision](https://via.placeholder.com/150)

*Figure 9: Precision and Recall rates for translation symmetry evaluation No. 1. Number triplet (a,b,c) means number of images used for each contestant. Here we used the only four images for which all contestants got valid results.*
We observe that overall our baseline algorithm performs best with a recall rate of 0.76 compared to 0.36 (Cai) and 0.19 (Kim). Due to the small number of test images used for this evaluation, any further discussion of performance characteristics seems insensible.

**Evaluation 2**

We evaluated each algorithm separately on its own set of valid output images. However, by doing so we need to be cautious about comparing the obtained performance results, because each algorithm is evaluated on a different set of test images.

![Precision and Recall rates for translation symmetry evaluation No. 2. Number triplet (a,b,c) means number of valid images considered for each contestant. Here each algorithm was evaluated independently on all its valid output results](image)

We observe that out of a total of 31 test images, Cai’s algorithm produced 21 valid output results, Wu’s algorithm produced 18 and our baseline algorithm produced 30 valid output results. In terms of both recall and precision the baseline algorithm performs significantly better than the two contestants.