LINEAR TIME SYMMETRIC AXIS SEARCH BASED ON PALINDROME DETECTION

Shaoheng Liang, Jiansheng Chen, Zhengqin Li, Gaocheng Bai

Department of Electronic Engineering, Tsinghua University, Beijing, 100084, China

ABSTRACT

We developed a method searching for the main symmetric axis in an image based on the SAX representation and a classical linear time palindrome algorithm. This method generates a curve outlining the axis by dynamic programming and produces a straight axis by RANSAC fitting. The computational complexity is $O(mn)$ on an $m \times n$ image, which is linear to the pixel number. This method can be extended to multiple symmetric axes detection. Comparing with state-of-the-art symmetry detection methods, our method has a comparable precision and is much faster.

Index Terms— Symmetric axis, palindrome, SAX

1. INTRODUCTION

Symmetry is a ubiquitous feature of various objects, natural or artificial, thus is of significance in recognizing such objects. Various symmetric detection methods have been proposed in the past two decades. Liu et. al. [1] mainly aimed at papercut-patterns. Their method searches potential symmetric axes exhaustively and compute correlations between original image and its reflection about the potential axes. The axes with relatively high correlation are considered correct ones. Two weaknesses limited the performance of this method. First, the computational complexity is high. Second, the per-pixel computation of correlation is sensitive to noise and lighting, which may not appear in synthetic images but is common in real images. Loy and Eklundh developed a method [2] based on scale-invariant features by using the direction and scale provided by feature key points. The content-aware method reduces the interference of noise and lighting, but the matching of key points is still time consuming. Some methods take advantage of transform domains. Keller and Shkolnisky [3] observed characteristic patterns of symmetric figure in pseudo-polar Fourier domain and denoted an algebraic approach to locate axes after recognizing such patterns. Sun and Si [4] indicated a process using orientation histograms with FFT accelerating gradient computation.

To evaluate the performance of state-of-the-art symmetry detection methods, Park et. al. [5] established a measurement and built a test data set consists of both synthetic and natural images. Two reflectional ([1], [2]) symmetry detection methods were evaluated. Park et. al. concluded that based on the speed and the precision of detection, [2] has the overall best performance, whose precisions are 92% for synthetic images with single axis and 84% for natural images with single axis, while [1] reaches 62% and 29% respectively. For images containing multiple symmetric axes, sensitivity drops drastically to no better than 43% for both methods. The time cost of these two methods are about 20-40 seconds for 530×531 and 800×600 images. In practice, the time cost of these methods are somewhat high, considering that symmetry detection is usually a pre-processing step in computer vision tasks. In this work, we propose a method of both low theoretical computational complexity and short practical running time, whose precision is comparable to the existing methods.

This work was supported by the National Natural Science Foundation of China (61101152), the Tsinghua University Initiative Scientific Research Program (20131089382), and the Beijing Higher Education Young Elite Teacher Project (YETP0104).

2. METHODOLOGY

2.1. Overview

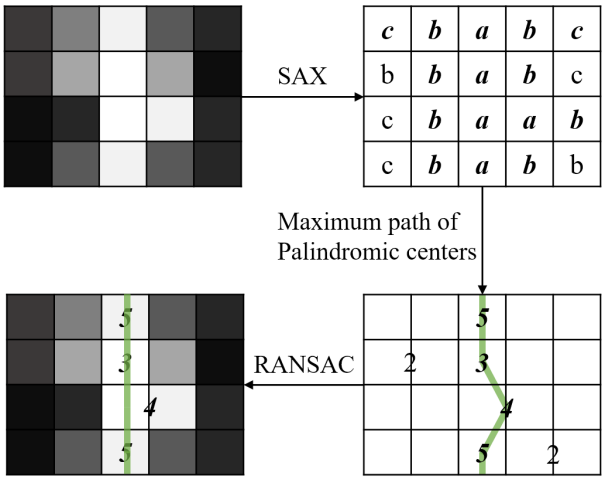


Fig. 1. Flowchart of our method. (For illustration only.)

"No 'x' in Nixon". Regardless of the spaces, punctuations and cases, this sentence remains the same if reversed. Such sentences are called palindromes. Surprisingly, all palindromic subsequences in a string can be found by a linear-time algorithm shown in section 2.3. Intuitively, palindrome is a

unidimensional case of reflection symmetry. When pick up a row of a mainly left-right symmetric image, it should contain a long palindromic substring. But, effected by lighting, noise, lossy compression, etc., the value of symmetric pixels may not remain identical, therefore the symmetry in images is not as strict as in strings. We will show an example in section 2.2. To suppress such influences, we use the SAX representation [6] to map various pixel values to characters in a small set before applying palindrome searching. Because most meaningful symmetric axes are nearly vertical, a horizontal radius for a point is sufficient. After marking each pixel with its palindromic radius, we look for a top-to-bottom path consisting of pixels with as large palindromic radii as possible for generating a proposal of a symmetric axis. We further use RANSAC to fit a straight line, and introduce a parameter to evaluate the confidence of the axis found. The flowchart of our method is shown in Fig. 1.

2.2. SAX Representation

Searching for palindromes on original images with various colors is highly sensitive to lighting, noise, lossy compression, etc. For instance, Fig. 3 is the enlarged view of area bounded by red rectangle in Fig. 2. Highly-symmetric as Fig. 2 is, the enlarged area is noisy and non-palindromic.

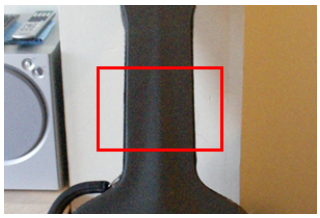


Fig. 2. Original Image.

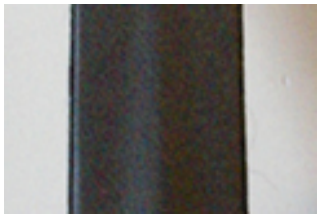


Fig. 3. Enlarged view.

A direct solution is mapping colors to a narrower palette, for instance 216 "web safe colors" [7], yet using the SAX [6] Representation based on normal distribution may be more robust. The SAX is a symbolic representation originally designed for time series data mining problems. To process image more effectively, Chen et. al. [8] proposed the 2D SAX image representation, under which the palindromic characteristic becomes more obvious as in Fig. 4. Palindromic centers and corresponding radii is shown in Fig. 5, where a symmetric axis lies clearly in the middle. The SAX representation can be done in linear time (i.e. proportional to pixel number).

2.3. Manacher Algorithm

The Manacher algorithm ([9], [10]) shown in Algorithm 1 is a linear time approach finding all palindromic substrings of a given string and marking every position in the string with its palindromic radius $r[i]$.
In brief, in line 11 we record the center id and right bound m

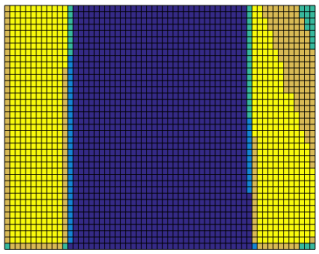


Fig. 4. the SAX representation.

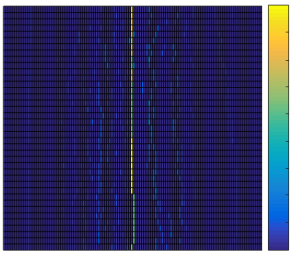


Fig. 5. Palindromic centers and corresponding radii.

Algorithm 1 Manacher Algorithm

Input: string $s[1 \dots n]$
Output: radius $r[1 \dots n]$

```
1:  r[1] ← 0
2:  id ← 0
3:  m ← 0
4:  for i = 2 ... (n − 1) do
5:      if i < m then
6:          r[i] ← min (m − i, r[2 ∗ id − i])
7:      end if
8:      while s[i + r[i] + 1] = s[i − r[i] − 1] do
9:          r[i] ← r[i] + 1
10:     end while
11:     if i + r[i] > m then
12:         m ← i + r[i]
13:         id ← i
14:     end if
15: end for
```

of the right-most known palindrome. Using this information, for $i < m$, the search of $r[i]$ starts with $\min(m - i, r[2 * id - i])$. Thus, there are three possible cases:

1. $m - i > r[2 * id - i]$, this implies that $r[i] \equiv r[2 * id - i]$. Thus, condition of the while loop at line 8 will not be satisfied at the very beginning, so line 9 will not be executed.

2. $m - i < r[2 * id - i]$. similarly, $r[i] \equiv m - i$. Line 9 will not be executed.

3. $m - i = r[2 * id - i]$, implies that $r[i] \geq m - i$. Certain value of $r[i]$ will be decided by the while loop. This is the only case that line 9 can be executed. Meanwhile id and m shall be updated.

For $i \geq m$, search of $r[i]$ start with $r[i] \leftarrow 0$, and id and m will be updated. The cost of this algorithm is determined by the execution times of line 9. It can be observed that this line will only be executed when m updates, and the execution times of line 9 equals the increment of m. Because m will stops at n, the execution time of line 9 will not exceed n, thus this is an algorithm of complexity $O(n)$.

Algorithm 1 suits only odd-length palindromes, but could be easily extended. To suit both odd-length and even-length palindromes, we insert an additional character between each pair of characters in s. As such, r becomes the entire length of the palindrome in original string. Risk of subscription out of range in line 9 can also be suppressed by adding two distinctive characters to the start and end of s.

There is one more issue to be solved when applying Algorithm 1 to symmetry detection. Let's consider a string "aaabbbaaa", in which the first and second occurrences of "aaa" are also palindromes. However, a more promising symmetric center should be the "b" in the middle. Therefore, we introduce Algorithm 2 to ignore substrings entirely overlaid by longer ones. Radii of the ignored palindrome will be set to zero. This algorithm maintains the furthest bound of

Algorithm 2 Select Palindromes

Input: length $r[1 \dots n]$
Output: selected position $p[1 \dots n]$
1: $p[1 \dots n] \leftarrow$ **true**]
2: $rightOverlap \leftarrow 0$
3: **for** $i = 1 \dots n$ **do**
4: **if** $i + r[i] \leq rightOverlap$ **then**
5: $p[i] \leftarrow$ **false**
6: **else**
7: $rightOverlap \leftarrow i + r[i]$
8: **end if**
9: **end for**
10: $leftOverlap \leftarrow n + 1$
11: **for** $i = n \dots 1$ **do**
12: **if** $i + r[i] \geq leftOverlap$ **then**
13: $p[i] \leftarrow$ **false**
14: **else**
15: $overlapBound \leftarrow i - r[i]$
16: **end if**
17: **end for**

scanned palindromes to discard latter palindromes entirely falls in it. This procedure runs for two times, back and forth, to find all overlaid palindromes. Consequently, the complexity is also $O(n)$. Number of rows m considered, the overall computational complexity of this step is $O(mn)$.

2.4. Axis Finding

Applying the SAX Representation and Manacher Algorithm on each line, we map original image to a matrix of "palindromic radii", denoted as $R_{m \times n}$ as in Fig. 5. Observed that the margin of images is usually background and shall not contain meaningful object symmetric axes, we weight each row of R with a raised cosine function. Similarly to seam carving [11], we use a classical dynamic programing technique as following to find the maximum weighted path from the top to the bottom of R. More specifically, we look for an array of $\{j_i \mid 1 \leq i \leq n\}$, satisfying equation (1), condition that $|j_i - j_{i-1}| \leq s$, in which s, the searching range of j_i's, is a small constant.

$$j_i = \arg\max_{j_i} \sum_{i=1}^{n} R(i,j) \tag{1}$$

Denote $opt(i,j)$ as the partial maximum weighted path ended at (i,j), the state transition equation of m is equation (2).

$$opt(i,j) = \begin{cases} R(i,j) & \text{if } i = 1, \\ \max_{|k-j| \leq s} opt(i-1,k) + R(i,j) & \text{otherwise.} \end{cases} \tag{2}$$

which can be solved in $O(mn)$. The maximum weight path can be rewrite as equation (3).

$$\sum_{i=1}^{n} R(i,j_i) = \max_{j} opt(m,j) \tag{3}$$

This dynamic programming problem can be solved in $O(n)$, and j_i's can be found by transition information in $O(m)$. Thus, the overall time complexity of this step is $O(mn)$.

2.5. RANSAC Fitting

The path found is not necessarily straight, especially when the main object occupies only a part of the image. To get a straight line, we use RANSAC fitting to discard outliers, and find the shortest segment containing most (for example 80%) of the remaining points, based on which the final result is generated using linear fitting. The ratio of non-discarded points, denoted r_G, can be regarded as the confidence of the found axis. The higher r_G is, the more likely a true axis is found. Zuliani [12] pointed out that RANSAC is an indeterministic algorithm whose computational complexity is related to the iteration times in equation (4), in which m_I is the number of inner points and ε and k are constants of the fitting model.

$$T_{iter} = \lceil \frac{\log \varepsilon}{\log\left(1 - (m_I/m)^k\right)} \rceil \tag{4}$$

For searching the axis on an obvious object, the inner point probability m_I/m shall be in a range regardless of the image height m, i.e. T_{iter} is nearly a constant. Thus, the overall complexity of RANSAC is shown in equation (5), where $C_{estimate}(k)$ and $C_{fitting}$ are constants of the model.

$$T = O(T_{iter}(C_{estimate}(k) + mC_{fitting})) \approx O(m) \tag{5}$$

Due to above-mentioned analysis, the overall time complexity of our method is $O(mn)$, which is linear to pixel number.

3. EXPERIMENTS

In Fig. 6, the curve shows the maximum weighted path. We observed that in most images, parts of curve on the symmetric object are usually straighter than parts on the background.

Hence, the outliers can be easily thrown by RANSAC. The discarded parts are marked by red. We search for the shortest section containing a fixed ratio of remaining green points, and do linear-fitting on it to get the final result shown in Fig. 7.

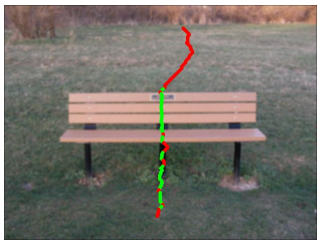


Fig. 6. Maximum weighted path.

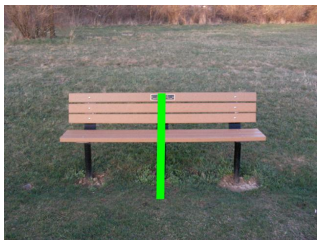


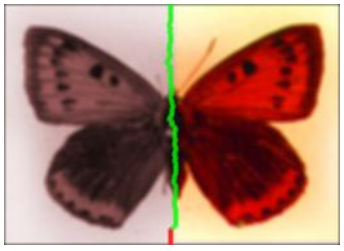
Fig. 7. Final result.

(a) Slightly non-verticle axis.

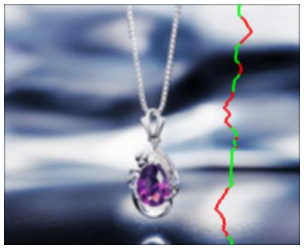
(b) Non-verticle axis.

Fig. 8. Examples for non-vertical axis.

Fig. 8(a) and Fig. 8(b) show that minor rotation of object has little effect on our method. The correct symmetric axis can still be found. Fig. 9(a) and Fig. 9(b) show the confidence of axis mentioned in section 2.5. A correct path should be straighter than a false one, thus is of higher r_G.



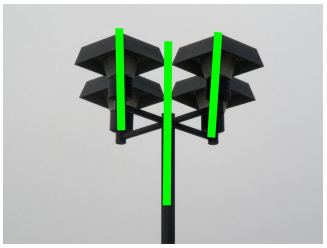
(a) Correct axis. $r_G = 0.9943$.



(b) Incorrect axis. $r_G = 0.6366$.

Fig. 9. Relationship between r_G and confidence of axes.

Our method has the potential for multiple symmetry detection. In Fig. 10(a), we shows a result generating by searching the first three non-intersect maximum weighted paths. On this image, this extension performs better comparing with method



(a) Our method.

(b) Method of Loy and Eklundh.

Fig. 10. Comparison on multiple axes detection.

of Loy and Eklundh as in Fig. 10(b). But determining whether an axis found is true or false is difficult. As is shown in Fig. 11(a), except for two vehicles, a meaningless false positive appears in the middle. By contrast, as in Fig. 11(b), method of Loy and Eklundh is less likely to produce false positives. To detect multiple symmetric axes, more effort is need for discarding false positives.

(a) Our method.

(b) Method of Loy and Eklundh.

Fig. 11. Fail case of multiple axes detection.

We run quantitative evaluation on test set of [5] using the single axis detection. The experiment is performed on a laptop PC equipped with a 2.7GHz Intel Core i7-5700HQ processor and 16GB memory using mixed programming of MATLAB and C++. The precision of our method reaches 92% on synthetic images with single symmetric axis, and 79% on real images with single symmetric axis, which is higher than [1] (62%, 29%), and slightly lower than [2] (92%, 84%, i.e. two more correct axes than our method). The time cost on 530×531 images is 0.18 second in average, i.e. about two orders of magnitude faster.

4. SUMMARY

In this work, we developed a practical method of low computational complexity detecting main symmetric axis in images. This method achieves comparable result with other two state-of-the-art methods, while performs much faster. Our method can be extended to multiple symmetric axes detection, while identifying false positives is still a challenging problem to be solved in the future.

5. REFERENCES

[1] Yanxi Liu, James Hays, Ying-Qing Xu, and Heung-Yeung Shum, "Digital papercutting," in *ACM SIGGRAPH 2005 Sketches*. ACM, 2005, p. 99.

[2] Gareth Loy and Jan-Olof Eklundh, "Detecting symmetry and symmetric constellations of features," in *Computer Vision–ECCV 2006*, pp. 508–521. Springer, 2006.

[3] Yosi Keller and Yoel Shkolnisky, "An algebraic approach to symmetry detection.," in *ICPR (3)*, 2004, pp. 186–189.

[4] Changming Sun and Deyi Si, "Fast reflectional symmetry detection using orientation histograms," *Real-Time Imaging*, vol. 5, no. 1, pp. 63–74, 1999.

[5] Minwoo Park, Seungkyu Lee, Po-Chun Chen, S. Kashyap, A.A. Butt, and Yanxi Liu, "Performance evaluation of state-of-the-art discrete symmetry detection algorithms," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, June 2008, pp. 1–8.

[6] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*. ACM, 2003, pp. 2–11.

[7] Michael H Birnbaum, *Introduction to behavioral research on the Internet*, Pearson College Division, 2001.

[8] Jiansheng Chen, Yiu-Sang Moon, Ming-Fai Wong, and Guangda Su, "Palmprint authentication using a symbolic representation of images," *Image and Vision Computing*, vol. 28, no. 3, pp. 343–351, 2010.

[9] Glenn Manacher, "A new linear-time "on-line" algorithm for finding the smallest initial palindrome of a string," *J. ACM*, vol. 22, no. 3, pp. 346–351, July 1975.

[10] Alberto Apostolico, Dany Breslauer, and Zvi Galil, "Parallel detection of all palindromes in a string," in *STACS 94*, pp. 497–506. Springer, 1994.

[11] Shai Avidan and Ariel Shamir, "Seam carving for content-aware image resizing," in *ACM Transactions on graphics (TOG)*. ACM, 2007, vol. 26, p. 10.

[12] M. Zuliani, *RANSAC for Dummies*, [online]. Available: http://old.vision.ece.ucsb.edu/~zuliani/Research/RANSAC/docs/RANSAC4Dummies.pdf.