2D Symmetry Recognizer and Beautifier

CS 8903 Special Problem: Final Report

Deep Ghosh
Georgia Institute of Technology
dghosh33@gatech.edu

1 Abstract

Symmetry detection and extraction is a concept that has received a lot of attention in the computer vision and computer graphics world. In the following report we present our efforts to automatically detect the most aesthetic pattern in a given 2D curve and once identified use the identified pattern to beautify the 2D curve. Current literature has been used to identify geometric patterns in 2D and 3D Geometry. We try in our effort to improve on these methods, in 2D, to detect any form of symmetry in a given curve and use the found element to beautify the curve in real time. We have constrained our research in the 2D domain and also on curves that are increasing in one axis. In our research, we look for ways to do the detection in real time. Going forward we will provide the mathematical model and/or definition for least square registration, dynamic time warping and curvature. We will present our algorithm using 2 different approaches and compare their performances, pros and cons.

2 Introduction

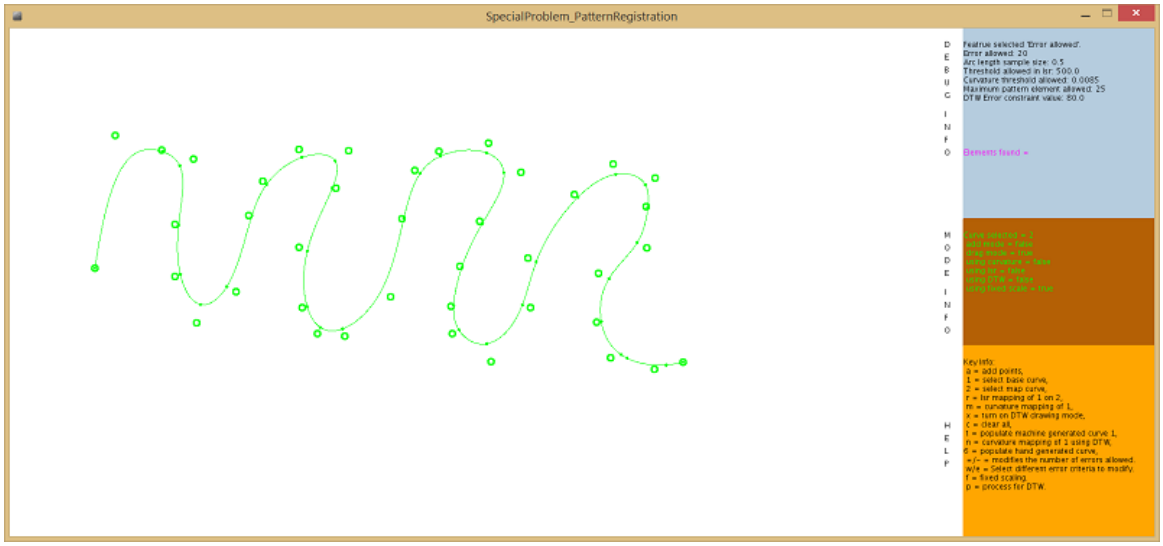

Figure 1: Sample Hand Drawn patterns.

Symmetry is ever present in nature and man made objects. Symmetry or repeated structures are present in every day life to various fields of physics, biology, manufacturing, arts. Symmetry is very important in different structures for reason's encompassing manufacturing efficiency, optimization, aesthetic purpose.
In the field of Computer Graphics, data corresponding to a particular shape, curve or model are captured in form of collection of low-level primitives. These primitives and the data structure storing them usually do not contain any information regarding any higher level of correlation between these

primitives. Lack of such information and given the importance placed on symmetries by various disciplines of science and arts has made symmetry detection a very important problem in the field of Computer Graphics. Symmetry information can be used to for different applications such as shape matching, segmentation, retrieval, beautification, procedural modeling etc.
The challenge in determining symmetry is presented by the lack of information(geometry, location, size etc.) regarding the shape being not present beforehand. The challenge significantly increases with the presence of patterns, patterns being present as part of larger geometric structure, outliers and noisy data. We present two different approaches pursued over the course of the semester to find symmetries in a constrained environment. The constraints defined for our algorithm are as follows,

- 2D curve.
- Curve always grows on x - axis.
- Curve drawn by user was drawn to have symmetry between its different parts
- Algorithm will detect only one pattern element in given curve, even when there are multiple possible pattern existing.

In [1],[**?**] and [2], we have a rich set of literature that has been discussed on a set of similar problems as to which we are pursuing. We found the paper [1] to be most relevant. We use concepts from [1], like creating a frequency map of the entire curve based on curvature to identify features of the curve and then using Least Square Registration to map the curve pieces. In the following paragraphs, we will go over the mathematical concepts and algorithm of Iterative Closest Point Matching, Curvature and Dynamic Time Warping. Following that we will show how we are leveraging these concepts in our two algorithms. Finally we will present our results along with a discussion on what we achieved as part of our research this semester.

3 Concepts

The following sections goes over the related concepts for our algorithm.

3.1 Iterative Closest Point Matching(ICP)

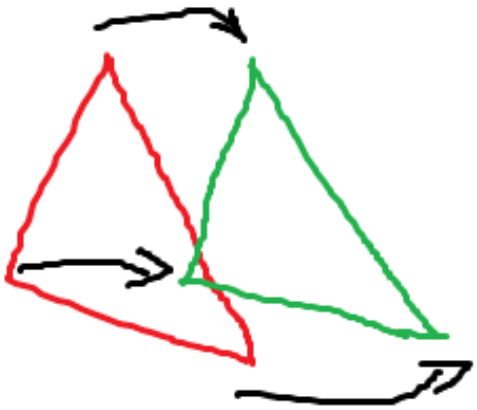

Figure 2: Iterative Closest Point Matching.

Iterative Closest Point Matching is a method of finding a similarity transform between two sets of points. Let us consider figure 2 for the purposes of understanding Iterative Closest Point Matching.

Looking at figure 2 we can see that there are 2 triangles , a red and a green one. Our objective is to transform the red triangle to a green triangle as per the black arrow's in the figure. The black arrows in the figure denotes the mapping of the corresponding vertex's for the transform. Iterative Closest Point Matching(ICP) is a method which determines a translation, rotation and scaling of the red triangle to a green triangle using a distance metric to minimize. The method used to minimize the distance between any two corresponding points in ICP uses Least Square Regression. The following equations are used for mapping the points for a similarity transform using ICP and Least Square Registration.

$$g(i) = dRr(i) + T$$

where r(i) stands for points on the red triangle. g(i) stands for the corresponding for r(i) on the green triangle. R stands for the rotation to be applied. T stands for the translation to be applied. d stands for the scaling to be applied. The Least Square Registration for the above equation will be of the following nature,

$$min(\sum((dRr(i) + T)^2 - g(i)))$$

The mapping transform will be taken to be the transform that reduces the overall least square distance gap between the 2 curves red and green.

$$T = g' - dRr'$$

where g' and r' are the center of mass of all the green and red triangle respectively.

$$d = \sqrt{\frac{\sum \mod (g(i) - g')^2}{\sum \mod (r(i) - r')^2}}$$

For rotation the following 3 equations gives the angle of rotation A

$$A = \begin{pmatrix} cos(a) \\ sin(a) \end{pmatrix}$$

$$A = \begin{pmatrix} \sum((g(i) - g').x * (r(i) - r').x + (g(i) - g').y * (r(i) - r').y) \\ \sum((g(i) - g').x * (r(i) - r').y + (g(i) - g').y * (r(i) - r').x) \end{pmatrix}$$

$$R = \frac{sin(a)}{cos(a)}$$

3.2 Curvature

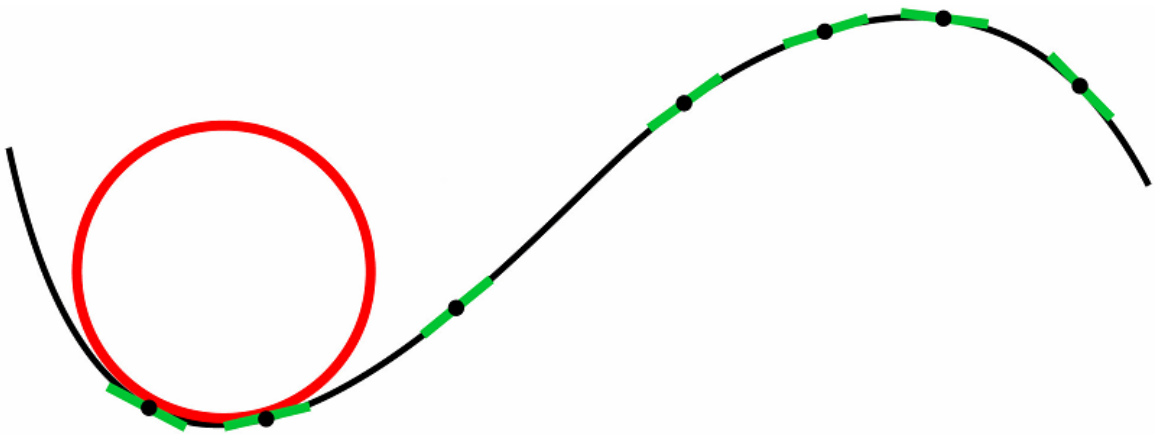

Figure 3: Osculating circle.

In our algorithm, curvature at any point of the sampled curve serves as feature. To measure curvature of any point of a curve we use the idea of an osculating circle. The curvature of a point is defined as the inverse of the radius of its osculating circle. An osculating circle at any point on a curve is defined as the circle passing through the point and its two adjacent neighbours on the curve.

3.3 Dynamic Time Warping(DTW)

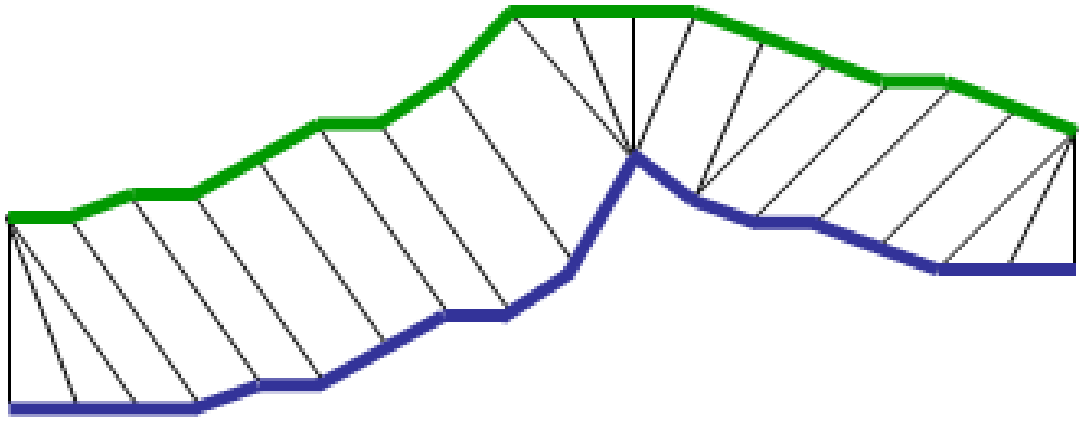

Figure 4: Dynamic Time Warping.

Dynamic Time Warping is a dynamic programming technique which is used in time series analysis to measure similarities between 2 sequences of different speed/time. Dynamic Time Warping creates a matrix between each and every point of the 2 curves in question using a distance metric. The distance metric is user defined. In our approach we are using the Least Square metric. Then the algorithm uses dynamic programming to search in the created matrix for adjacent cells with minimum metric starting from cell (0,0). This process is quite slow and curves with more than thousand points has huge performance penalties. There have been techniques developed in current literature which improves on the naive approach of DTW. [3] is the one of the more popular optimization techniques used to speed up DTW.

4 Overview of the proposed approach

In our application we make a comparative study between the 2 techniques ICP and DTW. For the comparative study we have made an application which allows the drawing of curves by drawing control points of a BSpline curve. Once the user draws a curve, the user can select a piece of the curve for comparison purpose using the 2 methods. We have implemented an auto registration feature as well using ICP, in which there is no need to select the second curve. DTW has not been used in the auto registration feature due to performance issues. Also DTW in it's original form compares between 2 signals. Hence it does take into consideration any sort of transformation as the signals are assumed to be starting from the same point. This is not true in our case. So our implementation of DTW is an amalgamation of cost mechanics of DTW and the similarity transform of ICP.

4.1 Outline of the algorithm

In our algorithm, we start by sub dividing the entire curve into regular samples based on arc length values. All of the components in our algorithm are performed on this newly sampled values rather than the original vertices. For auto detection and registration of curves, we start by taking the drawn curve and generating a frequency chart based on curvature values of each points on the sample space. The frequency chart contains the curvature value and their corresponding Genom class. The Genom class contains the count of occurrence of the curvature value, the index values of the sample for the said location and the sign of the curvature. We perform a descending order sort on the frequency chart. In the next step, we eliminate all anomaly data points in the chart by identifying too high or too low data points. Once the anomaly points have been rejected, we calculate the mean count value for all the curvature values present. We argue that a pattern has a higher chance of having all the curvature with occurrence count near the mean. Based on this logic we identify the largest

Data: curve
Result: Draw all identified elements on screen
arrCurv[] = generateCurvMap(curve);
arrCurv[].sort();
arrTrimCurv[];
for $i \leftarrow 1$ **to** $arrCurv.length$ **do**
    if $arrCurv(i) < thresholdValUpper$ **and** $arrCurv(i) > thresholdValLower$ **then**
        $arrTrimCurv.add(arrCurv(i))$;
    end
    Remove outliers from frquency map;
end
meanVal = getMinVal(arrTrimCurv);
$curStart = 0$;
$curEnd = arrTrimCurv(i).length() - 1$;
for $i \leftarrow 1$ **to** $arrTrimCurv.length$ **do**
    if $arrTrimCurv(i).cntVal < meanVal + thresholRegionVal$ **and**
    $arrTrimCurv(i) > meanVal - thresholRegionVal$ **then**
        $curStart = arrTrimCurv(i).getLowestVal$;
        $curEnd = arrTrimCurv(i).getHighestVal$;
    end
end
$elementFound = findAndRegister(arrCurv, curStart, curEnd)$;
return $elementFound$;

Algorithm 1: Detection and registration

curve defined by all the points with curvature in or near the mean, containing only one instance of each curvature value. Using the newly determined curve, we perform ICP to register and identify the pattern elements.

5 Results and Validation

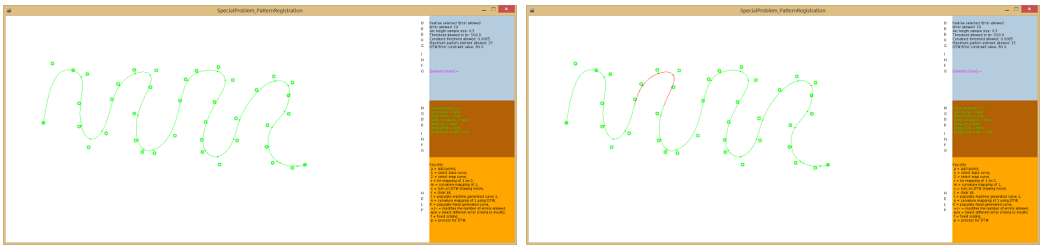

Figure 5: Initial Curve and Selected Curve.

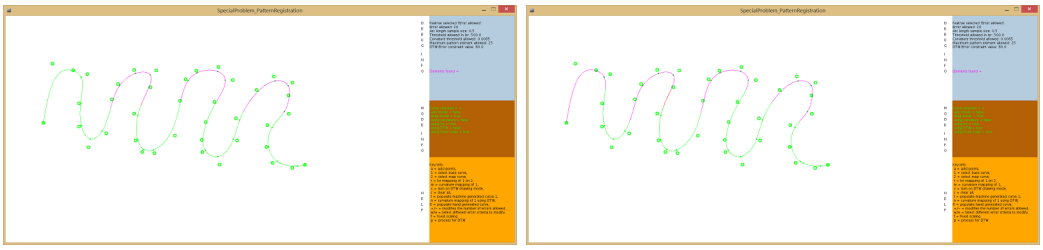

Figure 6: ICP matching result and DTW matching result.

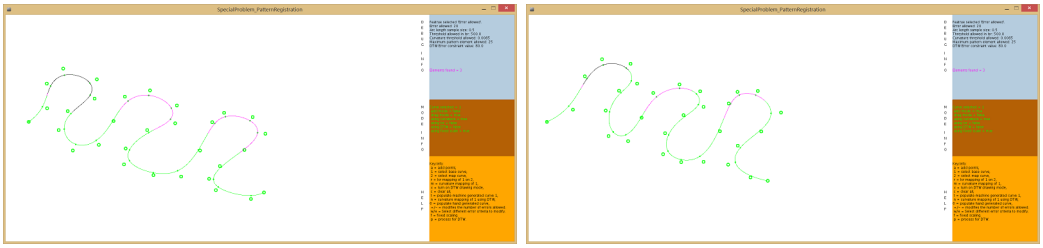

Figure 7: Auto detection(Good Result and Bad result).

Figures 5,6 and 7 are sample images generated from the application. The following section will discuss the results in greater details.

5.1 Performance

Performance comparison has to be looked at from 4 different perspective:

- Non Auto Registration ICP Algorithm: Performance was quite fast as the time complexity of the algorithm was worst case of O(N2).
- Non Auto Registration DTW Algorithm: Performance was significantly worse as the naive approach of DTW is quadratic in nature. On top of that we have modified the algorithm to include bunch of other performance intensive process for transforming the curve pieces and making sure that the entire curve is covered.
- Auto Registration ICP Algorithm: Performance is similar to Non Auto Registration ICP Algorithm as time complexity remains O(N2)
- Auto Registration DTW Algorithm: The functionality was not implemented as the cost in performance was too high.

5.2 Limitations

The algorithm is limited to curves growing on x-axis. Also in case of ICP the algorithm is very susceptible to scaling issues. The algorithm can handle curves which have patterns, which differ by small amounts. Larger anomalies cannot be handled by the application. Also due to the highly noisy nature of initial data, defining a pattern during auto registration is very susceptible to error.

5.3 Accuracy

Accuracy of the algorithm is varies. DTW has the best accuracy among the 3 approaches implemented. ICP sacrifices accuracy for performance. Auto registration ICP is very much error prone and there is no guarantee which pattern will be returned by this approach as evidenced in Figure:7. We can clearly see that a slight variation in the curve shape has given drastically different resulting output.

6 Conclusion

Our application showcased a very important observation that to get more accurate results, performance has to be sacrificed for existing algorithms. We also found that in case of a broad definition of curve, the data is very noisy and classifying them into buckets based on features introduces more errors in the process. Future work on this particular subject would require a identification of a technique to identify a feature that is error resistant. Once that is done a faster and more accurate method for mapping can be implemented to map the curves. In that regards FastDTW can be used.

7 Link to the code

`https://github.com/virtuahost/SpecialProblem_PatternRegistration.git`

References

[1] Johannes Wallner Helmut Pottmann Leonidas J. Guibas Mark Pauly, Niloy J. Mitra. Discovering structural regularity in 3d geometry. *ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2008*, 2008.

[2] Michael Wand Niloy J. Mitra, Mark Pauly and Duygu Ceylan. Symmetry in 3d geometry: Extraction and applications. *Computer Graphics Forum 2013*, 2013.

[3] Stan Salvador and Philip Chan. Fastdtw: Toward accurate dynamic time warping in linear time and space. *KDD Workshop on Mining Temporal and Sequential Data*, 2004.