Optimization of Point Selection on Digital Ink Curves

Rui Hu and Stephen M. Watt
Computer Science Department
University of Western Ontario
London, Canada
rhu8@uwo.ca, Stephen.Watt@uwo.ca

Abstract

Digital ink curves are typically represented as series of points sampled at certain time intervals. We are interested in the problem of how to select a minimal subset of sample points to approximate a digital ink curve within a given error bound. We present an algorithm to find an approximation with a specified number of points and providing the minimum cumulative error. Alternatively, it may be used to select the minimum number of points required to satisfy an error bound. The method uses dynamic programming and has a cost linear in the number of points.

1. Introduction

Pen input is one of the more convenient and natural forms of entry for several kinds of input, and has therefore been adopted for a variety of electronic devices such as tablets, PDAs, touch sensitive whiteboards, and cell phones. Tied to the pen input is the notion of digital ink. Digital ink is generated by sampling points from a traced curve at a certain rate, and thus is typically presented in the form of a series of points, each of which contains x and y values in a rectangular coordinate system at a particular time t. Recognition software applications take these sequences as input. In order to accommodate the need of detailed analysis and high definition rendering, higher sampling rate are used, allowing more points to be collected within an interval of time. However, this creates more work for recognition software applications and demands more resources for storage. We are therefore interested in how to select a subset of these sampled points that retain a desired degree of accuracy.

We are motivated by the problem of how to precisely approximate a digital ink curve through selecting a subset of points from the original trace. We wish to reduce the size of the subset while bounding the approximation's error. In other words, we would like to save the critical points that determine the shape of the curve (e.g. turning points) and remove those which have little impact (e.g. middle points on a straight line).

This is a problem for which there has been considerable previous work, some of which we highlight here. In 1986, Dunham [2] proposed an optimal algorithm to find a piecewise linear approximation with fixed initial and final points by selecting a subset of points from the original set. The approximation is optimal in the sense that it contains the minimum number of segments such that the error on each is below a uniform threshold. The error on each segment was taken to be the maximal distance from the curve segment to the line segment. In 1996, Horst and Beichl [3] introduced an algorithm which used arc-chord length difference as the error. Compared to [2], this algorithm achieved lower complexity but cannot guarantee global optimality. In 2007, another algorithm was presented in [5], which iteratively computes chordal deviation—the distance between the original curve and its approximation. Points with the minimal distance are removed until the distance becomes larger than a threshold. In 2012, Mazalov and Watt [6] described a piecewise linear approximation algorithm to compress digital ink. That algorithm is fast but suboptimal and selects points using a combination of two error functions. All of these algorithms compute the error on each curve segment and attempt to minimize the maximum error. They do not minimize the cumulative error which reflects the approximation's global deviation from the original curve. Sometimes the maximum error on each curve segment can be small but the cumulative error large, which can produce global distortion.

Our method is based on the observation that, for piecewise linear approximation, removing sample points gives approximating curves of shorter arc length. Arc length discrepancy is additive and may be used as a proxy for other error measures. A continuous curve may be approximated by a piecewise linear function with vertices on the curve. Decreasing the arc length discrepancy by adding points to a piecewise linear approximation decreases all the usual error measures and cannot increase them.

We present an algorithm to find an optimal point selection to approximate a piecewise linear curve. It can be used in two ways:

- Given a digital ink curve consisting of $n \geq 2$ points and a specified number of points $2 \leq k \leq n$, the algorithm selects a subset of k points such that the arc length discrepancy between the approximation and the original curve is minimized.
- Given a digital ink curve and a bound on arc length discrepancy, the algorithm selects a subset of points of minimum number required to approximate the curve to within that bound. That is, no smaller subset of the original points can achieve the bound.

Both uses are globally optimal and can be applied to both open and closed, planar and space curves.

The method can be applied when other error measures are of interest. In this case, though fast and good, the point selection is not guaranteed to be optimal. We have used this method with a variety of error types used in prior work, including the arc-chord length difference, maximal height, average height, which in turn measure the difference between the curve length and the chord length, the maximal height from the curve to the chord, and the average height from the curve to the chord. All of these errors are computed on each curve segment.

The remainder of the article is organized as follows. In section 2, we present the algorithm, its correctnes and complexity. Section 3 reports on experiments conducted to evaluate the performance of with several error functions. Section 4 concludes the article.

2 The Approximation Algorithm

2.1 Problem Definition

We consider a digital ink curve to be a two dimensional curve made up of a series of points. Our objective is to find an acceptable approximation by selecting a subset of points from the original ones. We have two problems: 1) Given a digital ink curve consisting of $n \geq 2$ points and a specified number of points k, $2 \leq k \leq n$, how can we select the k points such that the cumulative error between the approximation and the original curve is minimized? 2) Given a digital ink curve consisting of $n \geq 2$ points and a cumulative error threshold $\epsilon \geq 0$, how can we select the minimum number of points required to approximate the curve such that the cumulative error is less than or equal to ϵ?

Algorithm 1: Approximation by k points

```
Input: A digital ink curve of $n$ points, $n \geq 2$
Input: The specified number $k$, $2 \leq k \leq n$
Output: The indices of the k points
begin
    // The indices of the k points
    $S \leftarrow \{\}$;
    // The minimum weight table
    $D \leftarrow (k+1) \times n\ matrix$;
    // Path
    $P \leftarrow (k+1) \times n\ matrix$;
    // Initialization
    for $j \leftarrow 1$ to $n-1$ do
        $D_{2,j} \leftarrow w(v_0, v_j)$;
        $P_{2,j} \leftarrow 0$;
    // Compute the rest of $D$
    for $m \leftarrow 3$ to $k$ do
        for $j \leftarrow m-1$ to $n-1$ do
            $min\_weight \leftarrow \infty$;
            for $i \leftarrow m-2$ to $j-1$ do
                $weight \leftarrow D_{m-1,i} + w(v_i, v_j)$;
                $prior\_vertex\_index \leftarrow 0$;
                if $weight < min\_weight$ then
                    $min\_weight \leftarrow weight$;
                    $prior\_vertex\_index \leftarrow i$;
            $D_{m,j} \leftarrow min\_weight$;
            $P_{m,j} \leftarrow prior\_vertex\_index$;
    // Restore the path
    $vertex\_index \leftarrow n-1$;
    for $i \leftarrow 0$ to $k-1$ do
        $S \leftarrow S \cup \{vertex\_index\}$;
        $vertex\_index \leftarrow P_{k-i,vertex\_index}$
    return $S$
end
```

Both problems can be seen as graph problems. Given a digital ink curve consisting of n points, we first assign an index to each point. A weighted DAG (directed acyclic graph) $G(V, E)$ can be constructed from these points, where

$$\begin{cases} V &= \{v_i \mid 0 \leq i \leq n-1\} \\ E &= \{(v_i, v_j) \mid 0 \leq i < j \leq n-1\} \end{cases} \tag{1}$$

The set V contains n vertices, with v_i corresponding to the i-th point p_i on the digital ink curve. The DAG will be constructed to have a unique source (vertex with no inbound edge) and a unique sink (vertex with no outbound edge). The source corresponds to the initial point p_0 and the sink corresponds to the final point p_{n-1}. The weight of each edge is defined as:

$$w(v_i, v_j) = errorFn(p_i, p_j) \tag{2}$$

The error function $errorFn(p_i, p_j)$ is given beforehand. It measures the approximation error on the curve segment determined by p_i and p_j. Different error functions can be applied to compute for different error types. Section 2.3 will explain the error types in detail.

The first problem is now equivalent to finding a path from the source to the sink consisting of k vertices with minimum total weight. The second problem is equivalent to finding the shortest path from the source to the sink such that the total weight is less or equal to the given threshold.

2.2 Algorithm

Both paths are guaranteed to exist and can be found using dynamic programming. Given a graph $G(V, E)$, we define a matrix D, where $D_{m,j}$ represents the minimum total weight of the path from the source, v_0, to vertex v_j including m vertices. Initially, we assign

$$D_{2,j} = \begin{cases} \infty & \text{if } j = 0 \\ w(v_0, v_j) & \text{if } 0 < j \leq n-1 \end{cases}$$

For $m \geq 3$, $D_{m,j}$ can be computed as:

$$D_{m,j} = \begin{cases} \min\limits_{m-2 \leq i < j} \{D_{m-1,i} + w(v_i, v_j)\} & \text{if } j \geq m-1 \\ \infty & \text{otherwise} \end{cases}$$

Therefore, finding the minimum cumulative error of a k-point approximation is simply to compute $D_{k,n-1}$, where k is the specified number of points and $n-1$ is the index of the final point on the original curve. The complete algorithm to select the k points is shown in Algorithm 1.

Similar to Algorithm 1, finding an approximation consisting of the minimum number of points such that the cumulative error is within a given threshold, ϵ, is achieved by exiting the loop with a `break` statement. We keep computing $D_{m,n-1}$ for $m = 2 \ldots n$ until we find the first m that makes $D_{m,n-1} \leq \epsilon$. For additive errors, the m is guaranteed to exist as the cumulative error decreases when more points are selected and reaches 0 when all points are selected. The complete algorithm is laid out in Algorithm 2.

2.3 Error Types

We construct digital ink curves using linear and cubic spline interpolation methods since they are commonly used in the area of digital ink rendering, handwriting recognition, and handwriting neatening. By selecting a subset of points, the approximation algorithm introduces differences between the approximation and the original curve. These differences introduce error. The error is measured on each segment (i.e. the interval between each pair of points on the curve), and we assign zero to the error on the segment formed by any two consecutive points. Errors can be cumulated along the approximated curve, which reflects the global deviation from the original one. As digital ink curves may be generated in different scales, we normalize each by its arc length in order to evaluate the error fairly.

Algorithm 2: Approximation by error threshold

```
Input: A digital ink curve of $n$ points, $n \geq 2$
Input: The error threshold $\epsilon$, $\epsilon \geq 0$
Output: The indices of the selected points
begin
    // The selected points
    $S \leftarrow \{\}$;
    // The minimum weight table
    $D \leftarrow (n+1) \times n\ matrix$;
    // Path
    $P \leftarrow (n+1) \times n\ matrix$;
    // The smallest m that makes $D_{m,n-1} \leq \epsilon$
    $m^* = 2$;
    // Initialization
    for $j \leftarrow 1$ to $n-1$ do
        $D_{2,j} \leftarrow w(v_0, v_j)$;
        $P_{2,j} \leftarrow 0$;
    if $D_{2,n-1} > \epsilon$ then
        // Compute the rest of $D$
        for $m \leftarrow 3$ to $n$ do
            for $j \leftarrow m-1$ to $n-1$ do
                $min\_weight \leftarrow \infty$;
                for $i \leftarrow m-2$ to $j-1$ do
                    $weight \leftarrow$
                    $D_{m-1,i} + w(v_i, v_j)$;
                    $prior\_vertex\_index \leftarrow 0$;
                    if $weight < min\_weight$
                    then
                        $min\_weight \leftarrow weight$;
                        $prior\_vertex\_index \leftarrow i$;
                $D_{m,j} \leftarrow min\_weight$;
                $P_{m,j} \leftarrow prior\_vertex\_index$;
            if $D_{m,n-1} \leq \epsilon$ then
                $m^* \leftarrow m$;
                break;
    // Restore the path
    $vertex\_index \leftarrow n-1$;
    for $i \leftarrow 0$ to $m^*-1$ do
        $S \leftarrow S \cup \{vertex\_index\}$;
        $vertex\_index \leftarrow P_{m^*-i,vertex\_index}$
    return $S$
end
```

As we are interested in minimizing the global deviation error, we choose error types based on three criteria. A good type of error should be computationally

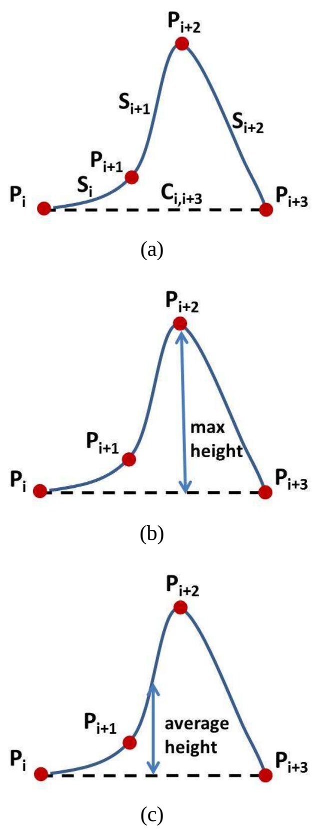

Figure 1. Error types: (a) Arc-Chord Length Error, (b) Maximal Height Error, and (c) Average Height Error. The curve is constructed using cubic spline interpolation.

efficient, additive, and has a natural meaning in geometry. In this article, we consider three types: Arc-Chord Length Error, Maximal Height Error, and Average Height Error.

- **Arc-Chord Length Error** measures the difference between the sum of the arc length of each curve piece and the length of the chord. An example is shown in Figure 1(a). The error on the segment (p_i, p_{i+3}) is computed as $S_i + S_{i+1} + S_{i+2} + S_{i+3} - C_{i,i+3}$, where S is the arc length of the curve piece and C is the chord length.

- **Maximal Height Error** measures the maximal distance between the curve segment and the line segment. An example is shown in Figure 1(b).

- **Average Height Error** measures the average distance between the curve segment and the line segment. An example is shown in Figure 1(c).

2.4 Correctness

Selecting the m-th point is a process of computing $D_{m,n-1}$, where $n-1$ is the index of the final point on the original digital ink curve. Since

$$D_{m,n-1} = \min_{m-2 \le i < n-1} \{D_{m-1,i} + w(v_i, v_{n-1})\},$$

we can recursively compute $D_{m,n-1}$, $m = 3 \dots n$ using dynamic programming with the given initial condition $D_{2,i} = w(v_0, v_i)$, $0 < i \le n-1$. This is a particular application of the Principle of Optimality [1] and $D_{m,n-1}$ will be the minimum cumulative error of the approximation consisting of m points.

Since the Arc-Chord Length error is additive, the matrix D has the following properties: with the increase of m, $D_{m,n-1}$ decreases and reaches 0 when $m = n$. But for the other two types errors, the matrix D may not have the property that $D_{m,n-1} \le D_{m',n-1}$ when $m' \ge m$. However, since $D_{n,n-1}$ is 0 in either case, we can always use the algorithm to find the solution.

2.5 Complexity

The complexity to find the k-point approximation is $O(kn^2)$. To see this, note we are computing a matrix. To select k poitns, it is necessary to compute k rows. Since each row has n entries, we have a total cost of $O(kn^2)$. Finding an approximation within a given cumulative error threshold ϵ, in the worst case that all points on the original digital curve need to be selected, giving complexity $O(n^3)$.

The both complexities can be reduced if we look at only a fixed number of prior vertices in computing $D_{m,j}$, but the approximation result may no longer be globally optimal.

3 Experiments

Figure 2 shows an example of applying Algorithm 1 to a digital ink curve. The digital ink curve consists of 55 points which are marked as black dots. The red dots are the selected points in the approximation. The error adopted here is the Arc-Chord Length Error. Increasing the number of selected points from 5 to 20, the approximation approaches the original digital ink curve quickly.

Figure 3 shows an example of applying Algorithm 2. The digital ink curve consists of 51 points which are marked as black dots. The red dots are the selected points in the approximation. The error adopted here is the Arc-Chord Length Error. As we decrease the error threshold ϵ, more points are selected in order to restrict the cumulative error within ϵ. When the error threshold drops to 0.001, the approximation is almost as the

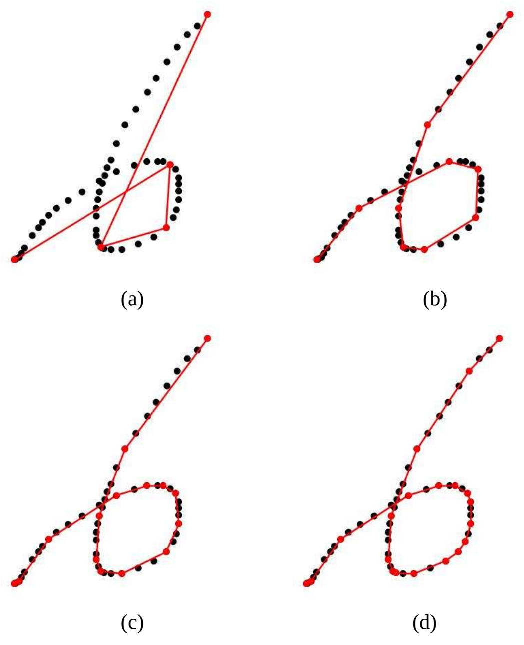
Figure 2. Approximation of symbol "6" by specified number of points k. The black points are the points on the original curve. The red points are the approximation. The error is the arc-chord length error. We use e to denote the cumulative error. (a) $k = 5$, $e = 0.052$. **(b)** $k = 10$, $e = 0.013$. **(c)** $k = 15$, $e = 0.005$. **(d)** $k = 20$, $e = 0.002$.

same as the original digital ink curve. But the number of points selected in the approximation is only half of the size of the original.

We have discussed three types of error in Section 2.3. To give a general idea of the performance for each error type, we have tested our algorithms against a handwriting dataset. The handwriting dataset we used is that of LaViola [4], containing 10665 symbols, mostly Latin letters and digits. Altogether there are 13203 strokes in these symbols. We constructed digital ink curves from these strokes using linear and cubic spline interpolation since they are commonly used in the area of digital ink rendering, handwriting recognition, and handwriting neatening. All of the curves were normalized by arc length in advance. We measured the average time cost in milliseconds and average cumulative error (relative to the trace length) on each digital ink curve. Figure 4 shows the average time cost of applying different error types in Algorithm 1. We see that the Arc-Chord Length Error outperforms the others in both linear and cubic spline cases. Figure 5 shows the average cumulative error of applying different error types in Algorithm 1. With the increase of the specified number of points, the average cumulative errors of all types drop dramatically, but takes longer to compute.

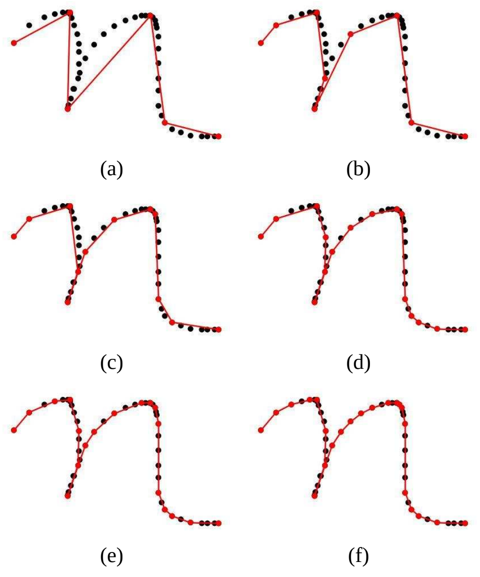
Figure 3. Approximation of symbol "n" by cumulative error threshold ϵ. The black points are the points on the original curve. The red points are the approximation. The error is the Arc-Chord Length Error. We use s and e to denote the number of selected points and the cumulative error, respectively. (a) $\epsilon = 0.05$, $s = 6$, $e = 0.0459$. **(b)** $\epsilon = 0.02$, $s = 9$, $e = 0.0182$. **(c)** $\epsilon = 0.01$, $s = 12$, $e = 0.0093$. **(d)** $\epsilon = 0.005$, $s = 16$, $e = 0.004$. **(e)** $\epsilon = 0.002$, $s = 19$, $e = 0.0018$. **(f)**$\epsilon = 0.001$, $s = 23$, $e = 0.0009$.

Evaluation of these error types can be conducted in different ways. Since the approximated curve and original curve share the same initial and final points, one can compare the area enclosed by the two curves. A smaller area indicates that the approximated curve in general looks more similar to the original one, which suggests a better approximation. When the enclosed area becomes 0, the approximated curve will overlap the original one and both curves will look exactly same. An alternative way to evaluate these error types is to compare the arc length between the approximated curve and original one. Since the two curves share the same initial and final points and our point selection is a subset of the entire point set, one can compute the difference between the arc length of the approximated curve and of the original curve. A smaller difference suggests a higher similarity and a better approximation. Figure 6 shows the average similarity which is defined as the arc length of the approximated curve divided by the arc length of the original curve. With the increase of the specified num-

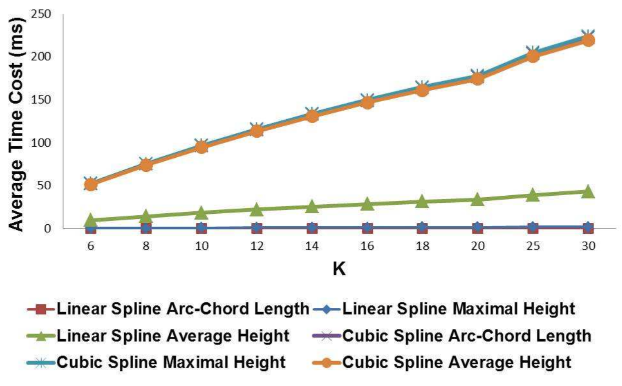

Figure 4. The average time cost of using different error types in Algorithm 1.

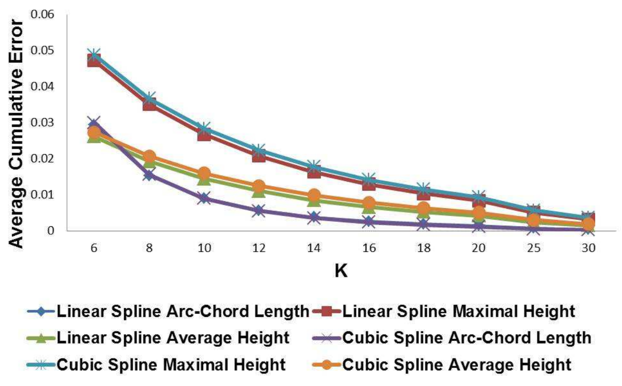

Figure 5. The average cumulative error of using different error types in Algorithm 1.

ber of points, the arc length of the approximated curves approaches the arc length of original curve quickly.

4 Conclusion

We have presented an algorithm to select a subset of points to optimally approximate a digital ink curve. In particular, it is able to find an approximation with a specified number of points and providing the minimum cumulative error or to select the minimum number of points required to satisfy a given error threshold. The algorithm is based on dynamic programming and has a cost linear in the number of points selected. The algorithm is independent of the choice of error function, and we have examined its performance with three: the Arc-Chord Length Error, the Maximal Height Error, and the Average Height Error. These were chosen for their computational efficiency and natural geometric meaning. Our experiments have shown that the Arc-Chord Length Error outperforms the others in terms of average time cost in both linear and cubic spline cases.

There are a few interesting directions we would like

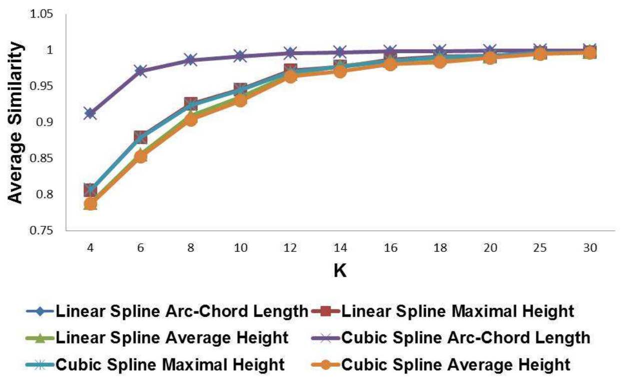

Figure 6. The average similarity between the approximated curve and original curve.

to pursue in the future. First, in addition to the cumulative error, we would like to restrict the error on each segment. This would allow control of the local deviation as well as the global deviation. Second, we wish to perform measurements with more types of error, including those involving differences in the spatial derivatives.

We would like to thank Enxin Wu, a Ph.D. candidate in the Department of Mathematics at the University of Western Ontario, for several useful discussions.

References

[1] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[2] J. G. Dunham. Optimum uniform piecewise linear approximation of planar curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(1):67–75, January 1986.

[3] J. A. Horst and I. Beichl. Efficient piecewise linear approximation of space curves using chord and arc length. *Proceedings of the SME Applied Machine Vision '96 Conference, Cincinnati Ohio*, June 3-6, 1996.

[4] J. LaViola. Symbol recognition dataset. http://pen.cs.brown.edu/symbolRecognitionDataset.zip.

[5] Z. Liu, H. S. Malvar, and Z. Zhang. System and method for ink or handwriting compression. *United States Patent No US 7,302,106 B2*, November 2007.

[6] V. Mazalov and S. M. Watt. Linear compression of digital ink via point selection. *10th IAPR International Workshop on Document Analysis Systems, Gold Coast, Australia*, March 27-29, 2012.