Parametric representation of digital shapes by Gaussian functions

A Goshtasby

A new formulation for parametric curves is given. In the proposed formulation, each component of a curve is defined independently of the other components and with different knots. A curve is represented by a blending of Gaussian functions. The Gaussian functions are estimated by the scale-space analysis of a digital shape. The estimated Gaussian functions are then refined by the Marquardt algorithm to minimize the root-mean-squared error between the curve and the shape. Numerical examples are given showing the accuracy and compression rate of the proposed parametric curve in the representation of digital shapes.

parametric curves, parameterization, digital shape, data compression, Fourier descriptors, scale-space images, Marquardt algorithm

A digital shape refers to a region boundary in a binary image. A binary image can be reconstructed if the boundaries of the regions in the image are known. In this paper, the parametric representation of digital shapes is discussed. The representation of digital shapes has been extensively studied. Many studies divide a shape into small pieces, and approximate each piece by a line segment such that an error criterion is met. Circular arcs and conic sections have also been used to represent digital shapes. The representation of digital shapes by line segments and curve segments has been mainly for data compression.

Digital shapes have also been represented by moments. Moments are particularly useful, because they may be made invariant with respect to the position, orientation, and scale of a shape. Moments have been used mainly for shape recognition.

A digital shape is given by the points \((x_k, y_k), \ k = 0, \ldots, N\)\). The shape is separated into \(x\) and \(y\) components in the form \(\{(t_k, x_k), \ k = 0, \ldots, N\}\) and \(\{(t_k, y_k), \ k = 0, \ldots, N\}\). \(t_k\) is the parameter value at \((x_k, y_k)\), and it is computed by \(k/N\). It is assumed that the first point is the same as the last point, and the digital shape is 4-connected (the distance between any two consecutive points is 1).

A parametric representation is useful for drawing...
A/Goshtasby

\begin{align*}
\delta x = x - t_j \delta t_j
\end{align*}

\textbf{Figure 1.} Shape: (a) digital shape, (b) x component of shape, (c) y component of shape

\{t \in \mathbb{R} \mid t \geq 0\}

\text{(reconstructing) a shape, because one can travel from one end of the shape to the other by changing the parameter from 0 to 1. A parametric representation is also compact, because only the parameters of the shape have to be saved. Usually, fixed blending functions are used to define parametric curves. Here, blending functions with free parameters are used to define parametric curves, and the free parameters are determined by minimizing the root-mean-squared error between the curve and the shape.}

The goal of this paper is to determine a set of Gaussian functions such that, when they are combined, an approximation to a digital shape is obtained with a required accuracy. The number of Gaussian functions required in this approximation depends on the detail of the shape and the required accuracy. In the following text, we express exponential of the form \( A \exp\left(-\frac{(t - t_0)^2}{2\sigma^2}\right) \) is referred to as a Gaussian function because it has the same shape as the Gaussian frequency function. \( A, t_0 \) and \( \sigma \) are called the amplitude, position and standard deviation, respectively, of the Gaussian function. A parametric curve in a plane is defined by \( \mathbf{P}(t) = [X(t) \ Y(t)] \), where \( X(t) = \sum_{j=1}^n A_j \exp\left(-\frac{(t - t_j)^2}{2\sigma_j^2}\right) \) and \( Y(t) = \sum_{j=1}^n A_j \exp\left(-\frac{(t - t_j)^2}{2\sigma_j^2}\right) \). Note that, in this formulation, the components of a curve are independent of each other, and have different knots. In the following text, a method that approximates each component of a digital shape by Gaussian functions with a required accuracy is described. Numerical results are given, and the compression rate of the Gaussian approximation is compared with that of the Fourier descriptors. In this paper, each component of a shape is referred to as a signal.

\section*{Approximation by Multicomponent Gaussian Functions}

Given a shape component by points \( \{(t_k, x_k), \ k = 0, \ldots, N\} \), it is required to determine a parametric curve in the form \( X(t) = \sum_{k=1}^N A_k \exp\left(-\frac{(t - t_k)^2}{2\sigma_k^2}\right) \) to approximate the points in such a way that \( \max|X(t_k) - x_k|; k = 0, \ldots, N < \varepsilon \). This problem can be transformed into that of solving a system of \( 3n \) nonlinear equations using \( (N + 1) \gg n \) data points.

A well known algorithm that solves a system of nonlinear equations is the Marquardt algorithm (see Reference 2 and Reference 4, pp 218–228). The Marquardt algorithm determines the nonlinear parameters of an approximating function by minimizing the sum of the squared errors between the approximating function and the given points. The algorithm is iterative, and it combines the Taylor-series expansion\(^2\) and the steepest-descent gradient method\(^2\) to achieve a fast convergence. Assuming that \( \mathbf{b} = [A_1, t_1, \sigma_1 \ldots A_n, t_n, \sigma_n] \) is the vector that represents the initial guesses of the parameters of \( n \) Gaussian functions, the Marquardt algorithm expands \( X \) by the Taylor series through the linear terms. That is,

\begin{align*}
\langle X(t_k, \mathbf{b} + \mathbf{d}) \rangle = X(t_k, \mathbf{b}) + \sum_{l=1}^{2n} \left( \frac{\partial X(t_k, \mathbf{b})}{\partial b_l} \right) d_l \\
 k = 0, \ldots, N
\end{align*}

or

\begin{align*}
\langle X \rangle = X_0 + \mathbf{Pd}
\end{align*}

where \( \mathbf{d} \) is a small correction to \( \mathbf{b} \), and the brackets \( \langle \cdot \rangle \) indicate linear prediction. Since Equation 2 is a linear function of the correction parameter \( \mathbf{d} \), it can be solved by the regular least-squares method. The sum of the squared errors is given by

\begin{align*}
E_2 = \sum_{k=0}^N [x_k - \langle X(t_k) \rangle]^2
\end{align*}

Setting \( \partial E_2/\partial d_l = 0 \) for \( l = 1, \ldots, 3n \), the following is obtained:

\begin{align*}
\mathbf{Ad} = \mathbf{c}
\end{align*}

where

\begin{align*}
\mathbf{A}^{[3n \times 3n]} = \mathbf{P}^T \mathbf{P} \\
\mathbf{p}^{[N + 1 \times 3n]} = \frac{\partial x(t_k, \mathbf{b})}{\partial b_l} \\
k = 0, \ldots, N, \ l = 1, \ldots, 3n \\
\mathbf{c}^{[3n]} = \sum_{k=0}^{N} (x_k - X(t_k, \mathbf{b})) \frac{\partial x(t_k, \mathbf{b})}{\partial b_l} \\
l = 1, \ldots, 3n
\end{align*}

\( \mathbf{T} \) is the matrix transpose operation. It has been shown that correction by \( \mathbf{d} \) does not always converge. To ensure convergence, an alternative method known as the steepest-descent gradient method must be used\(^2\). The gradient method makes the corrections in the direction of the negative of the gradient of \( E_2 \). The correction is given by

\begin{align*}
\mathbf{g} = -\left( \frac{\partial E_2}{\partial b_1}, \frac{\partial E_2}{\partial b_2}, \ldots, \frac{\partial E_2}{\partial b_{3n}} \right)
\end{align*}

The gradient method guarantees convergence, but it is
made to determine the Gaussian functions. A single Gaussian function produces two zero crossings, and there may be as many zero-crossing pairs in a signal as there are Gaussian functions in the signal. As the resolution of a signal is reduced, fewer zero crossings are obtained. Reduction in the resolution is achievable by smoothing the signal with a Gaussian filter (see Reference 29, pp 150–158). When reducing the resolution of a signal, zero crossings from small and noisy Gaussian functions disappear, and only zero crossings from dominant Gaussians remain.

When reducing the resolution of a signal, the positions of zero crossings may move. The positions of zero crossings at high resolution more accurately represent the characteristics of a signal. In a scale–space image, the presence of arches at low resolution show the existence of dominant Gaussian functions in the signal. By tracing the arches from low to high resolution, one can locate the positions of zero crossings that belong to dominant Gaussian functions in the signal. Tracing an arch from low to high resolution is possible because, when Gaussian filters are used to obtain a scale–space image, the resolution of the signal increases, an arch never disappears or branches into two.

Figure 2b shows the labeling of some of the arches. The arches are labeled starting at the lowest resolution (at the bottom of Figure 2b) and from left to right. Each arch has two arms. An arm of an arch is labeled by a positive number if the gradient of the signal at the corresponding zero crossing is positive. An arm is labeled as negative if the corresponding gradient is negative. If a Gaussian function has a positive amplitude, its negative arm appears to the right of its positive arm. When the negative arm of an arch appears to the left of the positive arm, it means that the underlying arch belongs to a Gaussian function with a negative amplitude.

The existence of zero crossings in a signal is due to the presence of two inflection points in each Gaussian function in the signal. The distance between the inflection points in a Gaussian function is 2π. The position of a Gaussian function is at the midpoint between its zero crossings. Therefore, if the zero crossings with labels i and −i are at \( t_i \) and \( t_{-i} \), the position and standard deviation of the ith Gaussian can, correspondingly, be estimated by

\[
t_i = (t_i + t_{-i})/2
\]

(6)

and

\[
\sigma_i = |t_i - t_{-i}|/2
\]

(7)

In this manner, the positions and standard deviations of all the Gaussian functions in a signal can be estimated. Once the positions and standard deviations of the Gaussian functions have been estimated, the amplitudes of the Gaussians can be determined by the regular least-squares method. Assuming that \( m \) zero-crossing pairs are obtained as a result of the scale–space analysis, the amplitudes that minimize the root-mean-squared error between the \( m \) Gaussian functions and a given signal.
satisfy the following system of linear equations:

\[
\sum_{i=1}^{m} A_i \sum_{k=0}^{N} \exp\left(-\frac{(t_k - t_j)^2}{2\sigma_i^2}\right) = \sum_{k=0}^{N} x_k \exp\left(-\frac{(t_k - t_j)^2}{2\sigma_j^2}\right) \quad j = 1, \ldots, m
\]

(8)

Note that the amplitudes of the Gaussian functions are determined optimally, but the calculations are based on the estimates of \( \sigma_i \) and \( \sigma_j \), which are not optimal, and so the computed amplitudes are also only estimates. \( A_i, \sigma_i \) and \( \sigma_j \) are used as initial values for the Marquardt algorithm to determine their optimal values by minimizing the approximation root-mean-squared error.

Rather than the parameters of the \( n \) Gaussian functions all being used at one time for the Marquardt algorithm, they are used one at a time until either the required accuracy is achieved, or all the \( n \) Gaussian functions have been used. Using Gaussian functions one at a time has the computational advantage that, at the \( (j + 1) \)th iteration, the process starts with the parameters of the \( j \) Gaussian functions optimally determined from the \( j \)th iteration. The parameters of the \( j \) Gaussian functions require only minor adjustments at the \( (j + 1) \)th iteration, and all that is needed is to determine the parameters of the \( (j + 1) \)th Gaussian function to achieve an optimal solution. The steps of the proposed algorithm are given below.

\[\text{Figure 3. Shape: (a) shape obtained by combination of Figures 3c and d, (b) approximation to Figure 1a with } \varepsilon = 3 \text{ using 12 Gaussian functions, (c) Gaussian approximation to Figure 1b with } \varepsilon = 8, \text{ (d) Gaussian approximation to Figure 1c with } \varepsilon = 8\]

**Algorithm**

**Fit algorithm**

Given \((N+1)\) uniformly spaced single-valued data, this algorithm approximates the data by a combination of Gaussian functions with accuracy \(\varepsilon\).

**Input:** Data points \(\{(t_k, x_k), k = 0, \ldots, N\}\) and accuracy \(\varepsilon\).

**Output:** A number \(n\) and parameters \(\{A_i, t_i, \sigma_i, i = 1, \ldots, n\}\) of Gaussian functions such that \(\max\{|\sum_{i=1}^{n} A_i \exp\left(-\frac{(t_k - t_i)^2}{2\sigma_i^2}\right) - x_k|: k = 0, \ldots, N\} < \varepsilon\).

**Method:**

1. Determine the scale-space image of the data, and label the arches in the image from low to high resolution. Suppose that \(m\) arches are obtained.
2. Estimate the parameters of the \(m\) Gaussian functions approximating the data using the formulae in Equations 6–8.
3. \(j = 0\).
4. If \(j \geq m\), then stop. Otherwise, use the parameters of the \(j\) Gaussian functions from the previous call to the Marquardt algorithm (if \(j = 0\), no previous parameters exist), and add to it the parameters of the \((j + 1)\)th Gaussian function that were computed in Step 2. Then, call the Marquardt algorithm using these parameters as the initial values.
5. \(j = j + 1\).
6. Compute

\[E_x = \max\{|\sum_{i=1}^{n} A_i \exp\left(-\frac{(t_k - t_i)^2}{2\sigma_i^2}\right) - x_k|: k = 0, \ldots, N\}\]

If \(E_x < \varepsilon\), then stop; otherwise, go to Step 4.

If the algorithm stops in Step 4, it means that the \(m\) Gaussians that were determined by the scale-space analysis are not sufficient to produce the required accuracy. To determine the remaining Gaussian functions in the data, the Gaussian functions that have been obtained so far are subtracted from the original data, and the residuals are used as new data. The new data is again decomposed using the Fit algorithm. This process is repeated until the required accuracy is achieved.

The Fit algorithm picks the Gaussian functions in the ascending order of their arc labels in a scale-space image. The arches are labeled in the order in which they emerge in the scale-space image while the resolution is increased. This ordering always picks the next most ‘dominant’ Gaussian function in a signal at each iteration.

Applying the data of Figures 1b and c to the Fit algorithm with \(\varepsilon = 8\), the approximations of Figures 3c
and d, respectively, were obtained. Two and four Gaussian functions were needed to construct Figures 3c and d, respectively. The shape reconstructed from Figures 3c and d is shown in Figure 3a. Using nine Fourier descriptors with \(\varepsilon = 13\), the shape of Figure 4a was obtained. Note that the numbers of scalars used to construct Figures 3a and 4a are the same. If the error measure is reduced to \(\varepsilon = 3\), the Gaussian method produces the shape of Figure 3b, with 12 Gaussian functions. Using 18 Fourier descriptors, the shape of Figure 4b was obtained, with \(\varepsilon = 6\). \(\varepsilon\) is the maximum vertical distance between each component of the shape and the approximating curve. The maximum distance between a shape and its approximation, therefore, cannot be greater than \(2^{1/2}\varepsilon\).

**Figure 4.** Approximations to shape of Figure 1a: (a) with nine Fourier descriptors producing \(\varepsilon = 13\), (b) with 18 Fourier descriptors producing \(\varepsilon = 6\)

<table>
<thead>
<tr>
<th>Shape number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Gaussian functions</td>
<td>14</td>
<td>15</td>
<td>6</td>
<td>16</td>
<td>12</td>
<td>9</td>
<td>8</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Number of Fourier descriptors</td>
<td>30</td>
<td>36</td>
<td>6</td>
<td>25</td>
<td>25</td>
<td>16</td>
<td>16</td>
<td>18</td>
<td>28</td>
</tr>
</tbody>
</table>

**Table 1.** Number of Gaussian functions and Fourier descriptors needed to approximate shapes of Figure 5 with \(\varepsilon = 3\)

<table>
<thead>
<tr>
<th>Shape number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\varepsilon) for Gaussian method</td>
<td>2.93</td>
<td>2.89</td>
<td>2.93</td>
<td>2.95</td>
<td>2.91</td>
<td>2.99</td>
<td>2.83</td>
<td>2.81</td>
<td>2.97</td>
</tr>
<tr>
<td>(\varepsilon) for Fourier method</td>
<td>5.72</td>
<td>6.21</td>
<td>2.85</td>
<td>5.30</td>
<td>2.85</td>
<td>4.92</td>
<td>5.23</td>
<td>4.32</td>
<td>8.92</td>
</tr>
</tbody>
</table>

**Table 2.** Accuracy of Gaussian and Fourier methods in approximation of shapes of Figure 5 with same data-compression rate

**RESULTS**

It is not possible to reach a general conclusion as to the superiority of the Gaussian representation over the Fourier descriptors. However, in the experiments carried out on different digital shapes, in most cases, the Gaussian representation performed better than the Fourier descriptors in data compression.

Figure 5 shows nine digital shapes that were obtained by segmenting an image of mechanical tools laid on a table. Each shape was decomposed into \(x\) and \(y\) components, and each component was decomposed into Gaussian functions. Table 1 shows the number of Gaussian functions and the number of Fourier descriptors needed to approximate each shape of Figure 5 with \(\varepsilon = 3\). Table 2 compares the accuracy of the Gaussian representation and the Fourier descriptors when using the same number of scalars to approximate the shapes of Figure 5. In Table 2, first, the number of Gaussian functions needed to represent a shape with \(\varepsilon = 3\) is determined (see the second row of Table 2). The actual error is shown in the second row of Table 2. Then, Fourier descriptors equal to 1.5 times the number of Gaussian functions are determined to approximate each shape. The error between a shape and its approximation is then determined and entered into the third row of Table 2. Therefore, the same number of scalars is used to reconstruct a shape by both the Gaussian functions and the Fourier descriptors. From Table 1, it can be concluded that, for the shapes in Figure 5 (except for shape 3), the Gaussian representation outperforms the Fourier descriptors in data compression. From Table 2, it can be concluded that, with the same data-compression rate, the Gaussian representation reconstructs the shapes of Figure 5 more accurately (except for shape 3) than the Fourier descriptors do.

Although shape recognition lies outside the scope of this paper, it is of value to mention briefly that the parameters of the Gaussian functions representing a shape may be used to recognize the shape. In shape recognition, a measure of similarity between two shapes has to be defined. In shape recognition by Fourier descriptors, the correlation between the magnitudes of the Fourier coefficients of two shapes is used as the similarity between the shapes. When two shapes are represented by Gaussian functions, a 3D vector showing the correlations of the corresponding parameters of the two shapes may be used to represent the similarity between the shapes.

Suppose that two digital shapes are represented by Gaussian functions. The similarity between the shapes may be described by \(r = [r_x, r_y, r_z]\), where \(r_x\), \(r_y\) and \(r_z\) are, respectively, the correlations of the amplitudes, positions and standard deviations of the Gaussian functions representing the two shapes. The magnitude of
vector $r$ may be used as a single scalar to measure the similarity between the shapes.

The computational complexity of the Fit algorithm is mostly that of the Marquardt algorithm, and the computational complexity of the Marquardt algorithm depends on the initial parameters provided and the required accuracy. For the shape of Figure 1a, 1.3 ms was required on an IBM 381 computer to reconstruct the shape of Figure 3b. The Fourier method had a lower computational complexity, requiring 0.6 ms to reconstruct the shape of Figure 4b. The computation times needed to obtain the approximations of Table 1 by the Gaussian representation were from two to four times those of the Fourier descriptors.

CONCLUSIONS

In the past, the same set of knots have been used to describe the $x$ and $y$ components of a parametric curve. If interactive design is not the objective, one may choose different sets of knots for the two components of a curve. A parameterization that describes one component of a curve may not be able to describe the other component satisfactorily. By using different sets of knots, one could obtain a more compressed representation and a better behaved curve. A representation for parametric curves has been described that uses different set of knots for the two components of a curve. The knots of a curve have been determined automatically by making the curve approximate a given set of data with a minimum root-mean-squared error.

Numerical results show that (except for very round shapes) the proposed representation provides a higher compression rate than the Fourier representation when approximating digital shapes.

REFERENCES

2 Freeman, H 'Shape description via the use of critical points' Pattern Recognition Vol 10 (1978) pp 159–165


