A NEW FAST AND EFFICIENT CONFORMAL MAPPING BASED TECHNIQUE FOR REMOTE SENSING DATA COMPRESSION AND TRANSMITTAL

Dalila Megherbi, member IEEE, and Joseph. Lucente, student member, IEEE
University of Denver
Denver, Colorado

Abstract- In this paper we deal with a new technique for large data compression. Contour mapping of two dimensional objects is of fundamental importance in remote sensing and computer vision applications. We present extensive algorithms applied to polygonized, simply-connected contours and reproduce desired shapes using an innovative data compression technique based on conformal mapping. In a previous work[3,4], through a conformal mapping process, we demonstrated the ability to 1) recognize shapes, and 2) concisely represent shape boundaries using a set of polynomial coefficients derived in the mapping process. In this work we illustrate how these previous results can be applied to data compression. In particular, in the approach outlined herein, a syntactic representation is formed for polygon shapes whose representation we desire to extract and reproduce compactly. Additionally, we present a problem of concavity in shape boundaries and a proposed solution in which polygons are divided into convex subsets and reconstructed accordingly.

Index Terms- Remote Sensing, Contour matching, Data Compression, Shape recognition, polygon decomposition

1. Introduction

The ability to efficiently process large datasets in a computational environment is a challenging task. In remote sensing applications, such as Earth sensing systems or computer vision systems, an abundance of data in the form of images is likely to exist. It is the desire of those who manage and use the data from these systems to realize efficiency in data transmission, manipulation and processing.[1] There exists a need to efficiently reconstruct images from data that has been transmitted on a network from remote sensing systems. The amount of information that is the result of a data compression technique plays a key role in the speed and efficiency in the process of reconstruction of remotely-sensed images. In order to quickly and efficiently reconstruct images from their compressed data, we propose a method in which the coefficients derived from mapping the boundary of an image to the unit circle[3,4] comprise the compressed data that is needed to reconstruct the original image. In this process, image objects are represented as polygons. We develop algorithms whose inputs are the vertices of polygons that represent the boundaries of image objects whose data we wish to compress. These algorithms divide the polygons into subsets of purely convex polygons. The convex subsets are then mapped to the unit circle using a conformal mapping process[3,4]. In this mapping process, a set of coefficients is derived and used in the proposed reverse mapping process to reconstruct the original image. It is these coefficients that are used to realize the data compression.

In this paper, we focus on an approach that uses polygonized, simply connected regions. We limit ourselves to the two-dimensional *shape* of the region, and do not concern ourselves with other region attributes such as motion, texture and color.[2] We describe in detail our data compression algorithm, and provide results of its application to some real images. But more specifically, we provide detail into our proposed approach of data compression that is realized in the algorithms described herein.

Finally, we present a problem of concavity in the forward and inverse mapping of simply-connected regions, and our proposed approach to overcome this problem. For polygons with concavities on the boundaries, we have developed an approach that recursively identifies such concavities by returning convex subsets of vertices of the original polygonized region whose data we wish to compress. We provide results of the reverse mapping process of original shapes having such concave regions, and discuss the quality of these results.

2. Data Compression Using Conformal Mapping Processes: A Proposed Technique

In this section, we present a set of algorithms developed to extract specific attributes of the shape of an object whose data we wish to transmit in a compressed format. We start with a known set of vertices in a coordinate system. We demonstrate two different methods of data population to construct the data points representing the sides of the polygon. We identify convexity or concavity at the vertices, and divide the polygon into convex subsets, each of which is mapped to the unit circle[3,4]. We extract the set of coefficients that are derived in the mapping process for later use in the shape reconstruction phase. Through observation of vertices of convex subsets between which there exist data which may or may not belong to sides in the original shape, we seek a data structure which holds indices to vertices belonging to the original polygon. In this data structure, there is a clear discernment between what we will show as *native* vertex links (i.e., data that belongs to the original shape), and *artificial* links (i.e., data that belongs to sides of convex polygon subsets, but is not part of the original polygon). We will demonstrate that this discernment is key to the successful reconstruction of the original polygon shape. Object rotation, translation, scaling and vertex convexity/concavity are shown to be invariant in these processes.

2.1. Preliminaries

In the algorithms developed for our proposed data compression techniques, a convention for the spatial ordering of vertices in all polygon subsets has been chosen to be a *counterclockwise* ordering in the Cartesian coordinate system.

We describe an important spatial relationship between vertices in our proposed algorithms as *vertex adjacency*. Two vertices are said to be *adjacent* to one another if no other vertices exist between them. In the discussion herein of spatial relationships of vertices, we present a symbolic convention for describing such adjacency between two vertices v_a and v_b as

$$v_a \leftrightarrow v_b$$

to indicate that v_a is immediately adjacent to v_b in some polygon P, in that no polygon vertices exist spatially between v_a and v_b. As will be seen, it is possible that two vertices v_a and v_b may be immediately adjacent in P, but may not necessarily be immediately adjacent in Q, where Q may be a convex subset of P. The usage of this symbolic convention is evident in the details of the proposed algorithms.

2.2 Compression Algorithms

The vertices of the object are first extracted from the image, and a syntactic representation is formed, the primitive elements of which are linear segments (the sides of the polygon). The attributes of the primitive elements are length, and orientation with respect to the coordinate system.[2] Our approach requires an identification of vertex concavity or convexity, as it is based on the mapping and subsequent reconstruction of the native links from the purely convex subsets of the original polygon. The algorithm for vertex labeling is outlined in section 2.3. We then proceed to the population of a data structure that holds the vertices of the convex subsets of the shape whose data we wish to represent in a compressed format. In the design of this algorithm, it is realized that complex polygonal shapes may contain regions of concavity in which their exist sub-regions of either additional concavity, or convexity. The division of such polygons into convex subsets demands a recursive process for the extraction of vertex subsets consisting of purely convex polygons, while preserving the spatial relationship between the vertices of each convex subset. Additionally, the chosen convention of counterclockwise ordering of the vertices in each convex subset is maintained within this process. Section 2.4 provides the details of this algorithm.

2.3 Labeling of Vertex Convexity and Concavity

The chosen approach for labeling vertices as convex or concave is inspired by the overall approach of dividing the polygonized object boundary into convex subsets. With the information as to which vertices are convex and which are concave, the foundation for the subsequent algorithm in which convex subset divisions are returned can be laid. A greedy labeling algorithm is proposed to minimize the number of convex subsets in P, while accounting for all vertices in the object boundary. The algorithm identifies a vertex v_m to be labeled as convex or concave by determining the angle β formed between adjacent vertices v_l and v_n, such that

$$v_l \leftrightarrow v_m \leftrightarrow v_n$$

in P. The label is applied to v_m according to the following convention.

$$\text{If } \beta < \pi,\ v_m \Leftarrow \textit{concave}$$

$$\text{If } \beta > \pi,\ v_m \Leftarrow \textit{convex.}$$

$$\text{If } \beta = \pi,\ v_m \Leftarrow \text{neither } \textit{convex} \text{ nor } \textit{concave}.$$

The algorithm is designed to begin at a vertex v_i of assumed concavity. The convention for identification of this vertex is to determine the vertex in P that holds the maximum Euclidean distance from c, the centroid of P. The algorithm proceeds in a counterclockwise direction from v_i until all vertices in P have been labeled. The pseudocode is provided as Algorithm 1.

2.4 Division of Object into Convex Subsets

A recursive process has been chosen for the extraction of sets of vertices in P comprising convex subsets. The resulting data structure is referred to in subsequent algorithms explained in sections 2.5 through 2.6. In Algorithm 2, the vertices of P have been labeled according to Algorithm 1. The process of vertex subset identification *Find_subsets (S)* begins with a null subset *convex_subsets* to which the set union of itself and the convex hull of P have been assigned. In the convex hull returned by the algorithm, vertices between which there exist any concavity form the starting and ending vertices of a set of cyclically-adjacent vertices in S, as a subset S_i of the convex_hull of P. Subset S_i is conditioned by reversing the convexity/concavity label at all vertices except the starting and ending vertices (which are convex in S), resulting in S'_i. Subset S'_i forms the recursive input as *Find_subsets(S'_i)*, the returned entity of which is the set of vertices comprising the convex hull of S'_i. Recursion proceeds until there are no pairs of vertices in P between which there exist concavity.

```
Let vertices be a cyclically-ordered set of $(x, y)$
coordinates of the $N$ vertices of a simply-
connected polygonized region $P$, where $N \geq 3$.

Let $c$ be the centroid of P.

Let $v_0$ be a vertex in $P$ whose Euclidean distance
from $c$ in $P$ is maximum.  By definition, $v_0$ is a
convex vertex in $P$.

Let $v_i$ be a vertex in $P$ for which a label of
convexity or concavity is to be determined.

For each vertex $v_i$ in vertices, where $i = 0, 1, 2, . .
., N-1$

    Let $vertices_{lmn}$ be a set of three cyclically-
    ordered vertices in $P$, such that $vertices_{lmn}$ =
    $\{ v_l, v_m, v_n \}$ where $vertices_{lmn} \subset V$ and
    $v_l \leftrightarrow v_m \leftrightarrow v_n$.  Let $v_m = v_i$.

    Let $\beta$ be an exterior angle in $P$ whose vertex
    is at $v_m$, and whose CW and CCW adjacent
    vertices are $v_l$ and $v_n$, respectively.

    If      $\beta < \pi$
        Label $v_m$ as concave
    Else If $\beta > \pi$
        Label $v_m$ as convex.
    Else If $\beta = \pi$
        Label $v_m$ as neither convex nor concave.
    End if
End for
```

Algorithm 2. Division of Object into Convex Subsets

Algorithm 1. Labeling of Vertex Convexity

```
Let SS be a cyclically-ordered set of $(x, y)$
coordinates of the $N$ vertices of a simply-
connected polygonized region $P$, where $N \geq 3$.

At each vertex $v_i$ (where $i = 0, 1, 2, . . . , N-1$) in
SS, let there be a label $L$ of convexity or
concavity, where $L_{vi}$ = concave or $L_{vi}$ = convex.

Let convex_subsets = { NULL }.

Find_subsets (SS, convex_subsets)
    Let $length_{SS} = N$.

    Let convex_hull_SS be the cyclically-ordered
    set of vertices in SS comprising the convex
    hull of SS.

    Let $l_{SS}$ be the number of vertices in
    convex_hull_SS.

        At each vertex $v_p$ (where $p = 0, 1, 2, . . . ,
        l_{SS} -1$) in SS, let convex_hull_SS contain
        a label of convexity or concavity.

        If $l_{SS} = = length_{SS}$
            convex_subsets $\Leftarrow$ convex_hull_SS
            return convex_hull_SS

        Else For each pair of vertices $p_f = ( v_a ,
        v_b )$ in convex_hull_SS, such that $v_a \leftrightarrow
        v_b$ in convex_hull_SS let $S_i$ be the
        complete set of $k$ cyclically adjacent
        vertices in SS such that $S_i = \{ vv_1, vv_2,
        vv_3, . . ., vv_k \}$, and $vv_1 \leftrightarrow vv_2 \leftrightarrow vv_3 \leftrightarrow . .
        . \leftrightarrow vv_k$ in SS, and $v_a \leftrightarrow \{ S_i \} \leftrightarrow v_b$ in
        SS.  Let $S_j$ be the set $\{ v_a ,\{ S_i \}, v_b \}$.

                If $S_i$ = = { NULL }
                    Skip $p_f$
                Else
                    For each vertex $vv_r$ in $S_i$
                      If $L_{vvr}$ is concave in SS
                        $L_{vvr} \Leftarrow$ convex in $S_i$
                      Else
                      If $L_{vvr}$ = = convex in SS
                        $L_{vvr} \Leftarrow$ concave in $S_i$.
                      End if
                    End For
                Find_subsets ($S_j$)
                End If
            End For
        End If
```

2.5 Mapping of Convex Subsets

In this section, we describe the steps we use to obtain a set of unique coefficients[3,4] for the geometric components of our original object, in the form of convex subsets. There are three parts to our research goal: 1) we wish to derive a set of coefficients to be used in the *unique identification* of the object irrespective of its translation, rotation, or scaling[3,4] 2) we wish to use the set of coefficients as a unique and concise *representation* of the object, and 3) we must be able to apply only the information used to represent the object to the *reconstruction* of the original object. This section describes the implementation of our first and second goals, where section 2.6 describes the proposed reconstruction algorithm.

In our proposed method, we divide our original object into convex subsets. We show our approach to the process of preserving the knowledge of which side polygonal segments are part of the original object, and which are not. We show how such preservation is made in the form of an interesting data structure, and how we use this data structure in the reconstruction phase.

Our approach to the derivation of convex subsets begins with vertex subsets which require a population of spatial points between each vertex. We propose two related methods to fill in data points, and, later in our results, we demonstrate the outcome of both methods.

We make reference to previous research in the application of conformal mapping techniques[3,4] in order to introduce our approach for deriving a unique set of coefficients for each convex subset. In later sections we apply this information to the reconstruction of the original object.

2.5.1 Division of Polygon into Convex Subsets

Convex subsets of our original shape boundary are represented as sets of polygonal vertices in an *(x,y)* coordinate system. Figure 1 shows an example of the division of a polygon exhibiting both concavity and convexity, into convex subsets. Notice the concave region consisting of vertices *a, b,* and *c*. In this region, Algorithm 2 returns a syntactical representation of this region in the form of

$$a \leftrightarrow b \leftrightarrow \mathrm{c} \leftrightarrow a$$

In comparison, notice the region consisting of vertices *e, f, g, h,* and *i*. Here, we see an example where Algorithm 2 will return recursively two convex subsets whose syntactical representation is

$$e \leftrightarrow f \leftrightarrow h \leftrightarrow i \leftrightarrow e$$

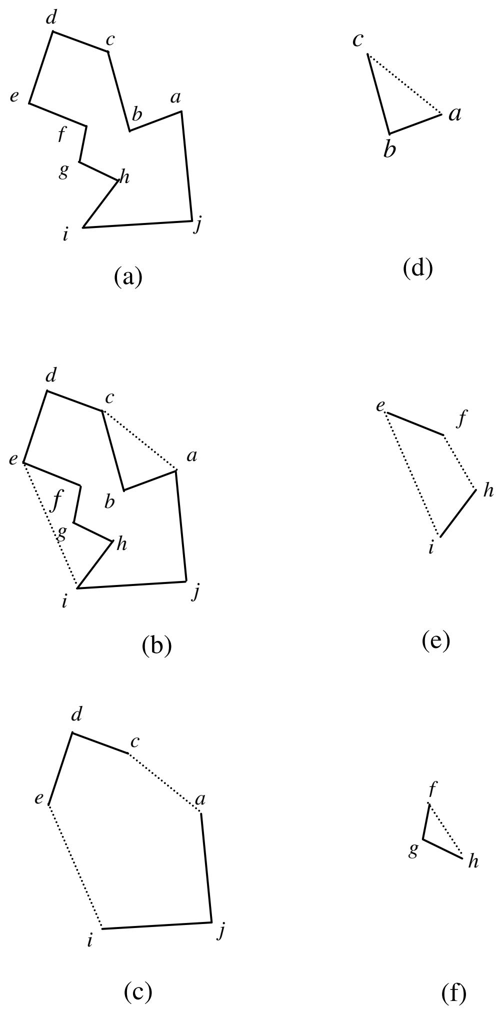
and

$$f \leftrightarrow g \leftrightarrow h \leftrightarrow i$$

With our original polygon having been divided into convex subsets according to Algorithm 2, we propose Algorithm 3 to prepare the convex subsets of *P* for subsequent mapping by creating data points between the vertices of the subsets returned by Algorithm 2. In Algorithm 3, data

Figure 1. (a) through (f) Convex Subsets of *P* where (a) shows original shape, (b) shows the convex hull of *P*, and (c) through (f) show the convex subsets of *P*. Native segments are shown as solid line segments, and non-native segments are shown as dashed line segments.

points are placed between each vertex in a give subset, according to a user-defined spatial resolution *step*. Algorithm 3 returns a data structure *shape* containing the vertices of the set of convex subsets returned by Algorithm 2, with data points, the Euclidean distance between which is *step*, placed between each pair of cyclically adjacent vertices. This algorithm simply calculates the Euclidean distance between each pair of cyclically adjacent vertices v_a and v_b, and divides



the distance by *step* to obtain the number of spatial data points that are needed between v_a and v_b to maintain a spatial resolution defined by *step.* Data points whose x and y coordinates are recorded in *shape* are then places between v_a and v_b. This process is repeated for all N cyclically adjacent vertex pairs.

Algorithm 3 populates cyclically adjacent vertex pairs with spatial data points in preparation for mapping of the convex subsets. Each subset S_i in P is represented syntactically by a set of cyclically-adjacent (x,y) coordinates in CCW order. For each pair of cyclically adjacent vertices (vs_a, vs_b) in S_i, *start* and *end* are assigned the (x,y) coordinates if vs_a and vs_b, respectively. The Euclidean distance between vs_a and vs_b is determined, and divided by *step* to establish the incremental spatial location of each data point between vs_a and vs_b. Upon completion, Algorithm 3 returns *shape*, consisting of all convex subsets of P, with data points between each pair of cyclically adjacent vertices. Each data point in *shape* is assigned an index. As indices are assigned to spatial points coincident with the vertices of the convex subsets, these indices are recorded in a separate data structure *links* which relates the vertices of the convex subsets with their *shape* indices. This step is important, as not all vertex pairs that are cyclically adjacent in a subset of P form native segments in the original polygon, as shown by the dashed lines in Figure 1.

We propose a slight variation of Algorithm 3 by altering the method by which the spatial location of data points is chosen. Unlike in Algorithm 3, where spatial location is chosen based on the Euclidean distance equal to *step*, in a direction along an imaginary line segment whose endpoint is the next cyclically adjacent vertex, the location of points chosen in Algorithm 4 is influenced not only by *step*, but also by proximity to the nearest vertex, and the magnitude of the interior angle at the nearest vertex. In Algorithm 4, datapoints are packed tightly near vertices, and more sparsely near midpoints between vertices. In no case, however, is the Euclidean distance between any two spatially adjacent data points any greater that *step*. A choice between Algorithm 3 and Algorithm 4 has consequences in the inverse mapping process, as will be demonstrated in our results. We present details of Algorithms 3 and 4 in Section 2.5.4.

Figure 2 shows, by contrast, a polygonal vertex at which data points are spaced equidistantly, as in Figure 2(a), and the same vertex with adaptive spacing of points. The equation for adaptive spacing of data points as shown by example in Figure 2(b) is

$$next\_step = step * (1/(1 + \text{distance})) * (\beta_m/\pi) \quad \text{(Eq.1)}$$

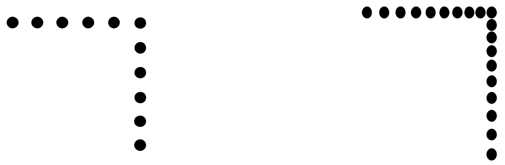


Fig. 2(a) Fig 2(b)

Figure 2(a) Equidistant spacing of data points in a typical polygonal vertex. Figure 2(b) Adaptive spacing of data points in a typical polygonal vertex

2.5.2 Maintaining Relationships Between Native Links and Artificial Links

In keeping with our goal to utilize a derive set of coefficients for both object *representation* and object *reconstruction*, we again stress the importance of maintaining a history of the specific data points which belong to native links and which belong to non-native links. We construct polygon subsets from our original concave polygon, and are forced to deal with links that do not belong to our original object. As stated earlier, we have shown such examples in Figure 1.

As we place data points between the vertices of each convex subset, we form a vector of (x,y) coordinates in an ordered fashion, such that the order of the data points follows the spatial ordering of the points in the convex subset. We label each vertex (x,y) coordinate with an index. Prior to data point population, whether by Algorithm 3 or 4, our *shape* vector contains only the original vertices from our convex subset, and their associated indices. As data points are inserted between vertices, the indices in *shape* associated with the original vertices will change. It is these changes in indices that we must understand.

We seek a data structure from which we may discern the nativity of vertex segments in the original object from the non-nativity of segments. We will ultimately use this discernment in the reconstruction of the original concave object in an *inverse mapping* process (see section 3.X) by acknowledging native segments and discarding artificial segments. Table 1 shows the *map* data structure for the polygon example in Figure 1, in tabular form. An important observation with respect to Figure 1 is that the table itself, implemented in the form of a data structure, does not maintain a history of vertex indices in the shape data structure. Rather, it tells us whether the data points between *original vertex indices prior to datapoint population* are (or are not) part of the original object. Once data points are placed between the vertices, we realize that indices for the original vertices will change. We maintain these changes in a separate data structure *links*, to which we relate the *map* data structure represented in Table 1. In the inverse mapping process, we will show how information is obtained from the *map* and *links* data structures to create a mask that is used to apply to the plotting of the data points for the reconstructed object.

Algorithm 4. Adaptive spacing of data points between vertices in polygon subsets

```
Let $R$ be a convex polygon with $N$ vertices, where N >= 3.

Let $DISTANCES$ = { NULL }.

Let $step$ be some maximum defined member in $DISTANCES$.

Let $l, m, n$ be the indices of three vertices in $R$ such that $v_l \leftrightarrow v_m \leftrightarrow v_n$ .

For $k$ = 1:N

        Let $\beta_m$ be the interior angle of $R$ at $v_m$ .

        Let $midpoint_k$ be one half of the Euclidean distance between $v_m$ and $v_n$ .

        Let distance = $midpoint_k$

        While $distance > 0$
                $next\_step = step * ( 1/(1 + distance) ) * ( \beta_m/\pi )$

                DISTANCES = DISTANCES $\cup$ $next\_step$

                $distance = distance - next\_step$
        End While

        $distance = 0$
        DISTANCE = DISTANCES $\cup$ $distance$

        Let $r$ be an index in $R$ such that $v_n \leftrightarrow v_r$ .

        Let $l = m$, $m = n$, and $n = r$.

        Let $\beta_m$ be the interior angle of $R$ at $v_m$ .

        While $distance < midpoint_k$
                $next\_step = step * ( 1/(1 + distance) ) * ( \beta_m/\pi )$

                DISTANCES = DISTANCES $\cup$ $next\_step$

                $distance = distance + next\_step$
        End While

        $distance = v_m$
        DISTANCE = DISTANCES $\cup$ $distance$
End For
```

2.5.3.1 Data Structure for Convex Subsets

Specifically, from Table 1 we may obtain the following information:

1. The number of convex subsets returned by Algorithm 2.
2. The specific vertices in each convex subset.
3. The cyclical adjacency of vertices in each convex subset.
4. The links between vertices in each convex subset that are native to the original polygon, and those that are not.

	a	***b***	***c***	***d***	***e***	***f***	***g***	***h***	***i***	***j***
1	√	*x*	√	√	√	*x*	*x*	*x*	√	√
2	√	√	√	*x*	*x*	*x*	*x*	*x*	*x*	*x*
3	*x*	*x*	*x*	*x*	√	√	*x*	√	√	*x*
4	*x*	*x*	*x*	*x*	*x*	√	√	√	*x*	*x*

Table 1. Map of the Vertices of the Four Convex Subsets of the Polygon in Figure 1. A √ indicates the vertex identifier in the corresponding column heading is a member of the set of vertices in the convex subset whose identifier is a row number. An *x* indicates non-membership in the subset of vertices in a given row. In any row, √'d vertex pairs between which there exist at least one *x* indicate non-native segments in *P*. Conversely, √'d vertex pairs between which there exist no *x* 's indicate native segments in *P*.

We may use the example of the polygon *P* shown in Figure 1, to validate the information in Table 1. We first observe four convex subsets from Figure 1 (c) through (f). Table 1 contains a row of vertices for each subset in *P*. In row number 1, we see the formation of a convex subset consisting of vertices *a, c, d, e, i and j.* Additionally, we see cyclical adjacency in this particular subset. As described in Table 1, we see the vertex pairs between which there exist at least one x, indicating non-native links in the original polygon. Information in the remaining three rows allows us to obtain similar information for the remaining convex subsets. This syntactic representation conveys the necessary information to reconstruct the relationship between the links in the polygon subsets and the links in the original polygon.

2.5.4.1 Details of Data Point Filling

In this section we present in detail our process for filling in spatial datapoints between the vertices of convex subsets. Recall that Algorithm 2 returns convex subsets in the form of a set of vertices S_i that are cyclically adjacent in S_i. We present the details of two methods for data point filling, and refer the reader to section 3.X for a comparison of some results of using these two approaches.

2.5.4.2 Equidistant Point Filling

Algorithm 3 populates cyclically adjacent vertex pairs with equidistantly spaced data points in preparation for mapping of the convex subsets. Each subset S_i in P is represented syntactically by a set of cyclically-adjacent (x,y) coordinates in CCW order. For each pair of cyclically adjacent vertices (vs_a, vs_b) in S_i, *start* and *end* are assigned the (x,y) coordinates of vs_a and vs_b, respectively. The Euclidean distance between vs_a and vs_b is determined, and divided by *step* to establish *nstep*, the number of data points between vs_a and vs_b. We recall that vs_b follows vs_a in a cyclically adjacent fashion, and forms a line segment as a side of S_i. As such, we form a vector $\boldsymbol{V}_{ab}$ whose endpoints are vs_a and vs_b. Using the direction $\boldsymbol{V}_{ab}$ derived from the x and y coordinates of *start* and *end,* and *step* expressed as a magnitude, we then obtain the dx and dy components of our desired data point k to create a (dx_k, dy_k) coordinate. We place this k^{th} coordinate into our *shape* data structure so that we maintain cyclical adjacency in *shape*, and repeat for all *nstep* points between vs_a and vs_b. Upon completion, Algorithm 3 returns *shape*, consisting of all convex subsets of P, with data points between each pair of cyclically adjacent vertices. Each data point in *shape* is assigned an index. As indices are assigned to spatial points coincident with the vertices of the convex subsets, these indices are recorded in a separate data structure *links* which relates the vertices of the convex subsets with their *shape* indices. This step is important, as not all vertex pairs that are cyclically adjacent in a subset of P form native segments in the original polygon, as shown by the dashed lines in Figure 1. Figure 2(a) shows an example of a cyclically adjacent set of equidistantly-spaced data points in close proximity to a vertex.

2.5.4.2 Adaptive Point Filling

Algorithm 4 populates cyclically adjacent vertex pairs with adaptively-spaced data points. In preparation for mapping of the convex subsets. We select the descriptive term *adaptive* due to the fact that a selection of any data point in this algorithm is influenced by some geometric characteristics of the polygon subset. We say that data point spacing *adapts* to the subset geometry.

Each subset S_i in P is represented syntactically by a set of cyclically-adjacent (x,y) coordinates in CCW order. As in Algorithm 3, we initialize *step* with a user-defined increment of resolution. For each of the N vertices in S_i we then select three vertices v_l, v_m and v_n in S_i such that

$$v_l \leftrightarrow v_m \leftrightarrow v_n$$

and

$$1 \le m \le N$$

We calculate β_m, the magnitude of the interior angle of $\boldsymbol{S_i}$ at v_m. We let $midpoint_k$ be the spatial location that is one half of the Euclidean distance between v_m and v_n. We let $distance_k$ be the Euclidean distance to $midpoint_k$, where

$$1 \le k \le N$$

We then calculate a Euclidean distance *next_step* from v_m according to Equation 1. We subtract from $distance_k$ the newly-calculated *next_step* from v_m. As this last step is repeated, $distance_k$ decreases with each addition of *next_step*. Our stopping condition for this segment of Algorithm 4 is when $distance_k$ is equal to zero. When this condition is true, we know that we have placed data points between v_m and $midpoint_k$.

We must now complete the adaptive placement of data points from $midpoint_k$ to v_n. In this last segment of Algorithm 4, we maintain knowledge of $midpoint_k$, but we select the *next* cyclically adjacent vertex r from v_l such that

$$v_n \leftrightarrow v_r \text{ and } r \ne m$$

We let $l = m$, $m = n$, and $n = r$. Thus we are selecting a set of three cyclically adjacent vertices in S_i that are "offset" in a CCW direction by one vertex from our most recent set of three vertices v_l, v_m and v_n. We calculate β_m, the magnitude of the interior angle of $\boldsymbol{S_i}$ at (new) v_m. We then calculate a Euclidean distance *next_step* from v_m according to Equation 1. We add to $distance_k$ the newly-calculated *next_step* from v_m. As this last step is repeated, $distance_k$ increases with each addition of *next_step*. Our stopping condition for this segment of Algorithm 4 is when the summed distance is greater than $distance_k$. When this condition is true, we know that we have placed data points between $midpoint_k$ and the (new) v_m.

We observe from Equation 1 that the distance *next_step* is a product of three multiplicands

1) *step*
2) $1/(1 + distance_k)$
3) β_m/π

As the summed distances $distance_k$ approach the distance between the nearest vertex and the midpoint of the line segment to which data points are being created, 2) becomes maximally influential to *next_step.* Conversely, as this summed distance approaches zero, 2) becomes minimally influential to *next_step.* In a similar fashion, 3) influences *next_step* as a ratio of the magnitude of the interior angle at the nearest vertex, to *pi*.

We structure the multiplicands in Equation 1 so as to effect a more compact placement of data points at vertices exhibiting relatively smaller magnitudes of interior angles. We place additional influence on

Algorithm 4. Inverse Mapping Algorithm (Part 1 of 2)

next_step by making it sensitive to proximity to midpoints. Spatial points are placed sparsely nearer to midpoints, and more densely nearer vertices.

2.6 Reverse Mapping of Convex Subsets

As described in the algorithms in section 2.3 through 2.5, a unique set of coefficients of convex subsets of polygon P is derived. We have included considerable effort to maintain a separation between the segments of each convex polygon subset, in order to ascertain the difference between native segments to P, and non-native segments. With this knowledge, and with a unique set of coefficients in z representing an object with N original data points that we wish to reconstruct, we posses the ability perform an inverse mapping to obtain the original image. Our motivation for selecting a mapping to the unit circle[3,4], as opposed to some other geometric object, are apparent in our goal to design a technology in which objects may not only be uniquely represented, but whose N data points may be

Algorithm 4. Inverse Mapping Algorithm (Part 2 of 2)

represented in a compact manner. This section describes how we make use of the set of unique coefficients to reconstruct the original object.

We choose the unit circle in the η plane due to its ease in construction as a preliminary step in the inverse mapping process. With a knowledge of N data points in the original object, we may construct a circle of radius r, where $r = 1$, with N data points on its boundary, each point of which may be represented as a complex number in the form of Eq. 2. We then propose a method by which we apply the set of n complex coefficients that we derived in the conformal mapping process. Our method is a straightforward approach which involves finding the roots of the polynomial at each point z_i in *shape*. As we know, a polynomial of degree p will produce a maximum of p roots. Thus we are forced with a choice of which of the p roots we wish to use in the inverse mapping process for each point z_i. We refer to[3,4] in a determination of this choice, and select the root with the minimum magnitude for each point z_i.

We propose Algorithm 5 as a solution to the inverse mapping problem. Complicated only by our attention to a record of discernment between indices indicating native segments in our original object, and indices indicating non-native segments, Algorithm 5 finds the minimum magnitude of all roots at each point on our construction of a unit circle in the η plane, and plots only those whose index points to segments whose spatial coordinates are represented between

```
Let count be the number of convex subsets in P.

Let n be the degree of a polynomial in z.

Let subset_coefficients be a set of coefficients
consisting of complex numbers in the $\eta$ plane
for a convex subset in P, given as

subset_coefficients = { $c_0, c_1, c_2, \ldots, c_n$ }

Let coefficients be the set of count sets of
coefficients subset_coefficients, where the kth
subset is given as

$subset\_coefficients_k$ = { $c_{k0}, c_{k1}, c_{k2}, \ldots, c_{kn}$ } where k = 1:count, and

coefficients = { $subset\_coefficients_1$,
$subset\_coefficients_2$, . . . ,
$subset\_coefficients_{count}$ }

Let native_links consist of the set of indices
of the data points in P between which there
exist points on native segments in P.

Let inverse_map = { NULL }.
```

```
For i = 1:count

        Let $N_i$ be the number of data points in
convex $subset_i$ .

        Let $delta\_theta_i = N_i/2\pi$.

Create a set of data points $circle_i$ in the $\eta$ plane,
consisting of $N_i$ complex points spaced equally
according to $delta\_theta_i$ , forming a circle of
radius r = 1.

For j = 1: $N_i$

    Let $index_i$ be the index of the root of the ith
    complex point in circle

    $\eta_j = z^0 + c_{j1}z^1 + c_{j2}z^2 + \ldots + c_{jn}z^n = z_j$

    $\eta_j = z^0 + c_{j1}z^1 + c_{j2}z^2 + \ldots + c_{jn}z^n - z_j = 0$

    $ROOTS\_minimum_i$ = minimum magnitude
    root of polynomial represented by $\eta_j$.

    If $index_i \in native\_links$

        $inverse\_map = inverse\_map \cup$
        $ROOTS\_minimum_i$

    End If

End For

End For
```

indices of native links in our original object. Data points derived from non-native links are rejected in the plotting phase of the reverse mapping process.

The reader may observe that previously discussed Algorithm 3 begins with a population of data points on convex subsets of the original object, spaced equidistantly. Algorithm 5 begins with the construction of points on a unit circle in the η plane. The choice of spacing of the points in η relies on a choice of *delta_theta*, so that the *N* data points on the original convex subset are equally spaced on the unit circle in η. In section 5, we discuss the consequences of such choices, and propose variations to Algorithms 3 and 5, with their results. For now, we concentrate on the results of Algorithm 5 when applied to polygons whose original data points are spaces equidistantly, and whose inverse mapping is derived from a unit circle consisting of equally spaced points in η.

3.0 Forward And Inverse Mapping Results

The next step in our approach involves feeding a set of vertices to our collection of proposed algorithms, and observing the results. We present results in two major categories: 1) forward mapping results, and 2) inverse mapping results. We wish to show results of forward and inverse mapping from an application of the approaches outlined in our explanation of Algorithms 3 and 4.

We include an example of the application of our proposed technique to a convex polygon, as shown in Figure 3(a) through (d). Figure 3(a) shows the original convex shape whose data we wish to compress. We decompose the original shape into concave subsets. For this example, the convex subsets consist of only the original convex shape itself, shown in Figure 4(b). For each concave subset, we obtain a set of coefficients using the techniques described in[3,4], the results of which are shown in Figure 4(c). We use these coefficients to represent the original dataset for the convex subset by feeding the coefficients into Algorithm 5 to produce the inverse mapping seen in Figure 3(d).

Figures 4(a) – (d) show the sequence of output images when we apply Algorithms 1, 2, 3 and 5 to produce an inverse mapping of on original polygon containing one concave region. The original 16 vertices are fed into Algorithm 1, resulting in a labeling of each vertex as *convex* or *concave*. The linear segments formed by vertex pairs are then populated with data points that are spaced equidistantly, by a chosen spatial resolution of 0.01 units. Figure 4(a) shows a plot of the data points produced by Algorithm 1. We then proceed to Algorithm 2, in which the simply-connected region in Figure 4(a) is divided into convex subsets. Figure 4(b) shows such a division. The two convex subsets are then feed into the algorithm discussed in[3,4]; this step results in a mapping to circles as shown in Figure 4(c). We have, at this point, a set of coefficients for each convex subset that has been derived in the mapping process. For the mappings shown in Figure 4(c), we have chosen a polynomial in *z* of degree 20, thus, we have a maximum of 20 coefficients per convex subset. We feed these sets of unique coefficients into Algorithm 5 to produce the inverse mapping seen in Figure 4(d).

4.0 Conclusions and Future Work

Current data compression schemes such as JPEG are capable of producing compression ratios between 10:1 and 20:1 for images without visible loss. JPEG compression ratios in the range of 30:1 to 50:1 are possible for images with small noticeable defects. For low-quality images (those which contain obvious noticeable defects) compression ratios as high as 100:1 are obtainable. [19] In our proposed technique, we demonstrate compression ratios of 101:1 and 123:1 with little to no visible defect, as shown in figures 3 and 4. These results demonstrate the power of the proposed technique over existing data compression methods. Future work in this area will include applications to non-polygonized, simply-connected regions.

5.0 Acknowledgements

Funding for this research is provided by the Rocky Mountain NASA Space Grant Consortium Fellowship. The authors would like to thank Dr. Jon Arvik for his comments and insight during this work.

Figure 3

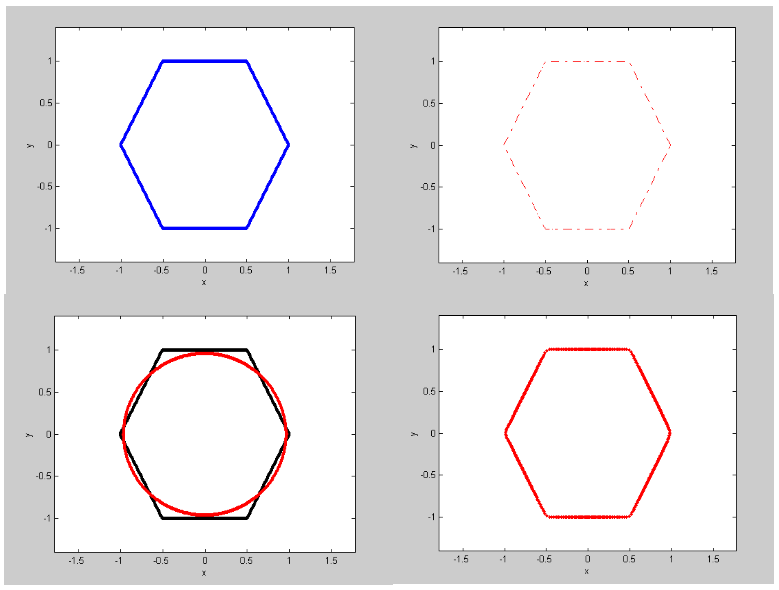


Figure 3(a) Convex polygon consisting of 2023 data points
Figure 3(b) Original shape decomposed into convex subsets (in this case, just one)
Figure 3(c) Original shape mapped to the unit circle
Figure 3(d) Reconstruction of the original shape through inverse mapping process, using 20 derived coefficients, resulting in a compression ratio of 101:1.

Figure 4

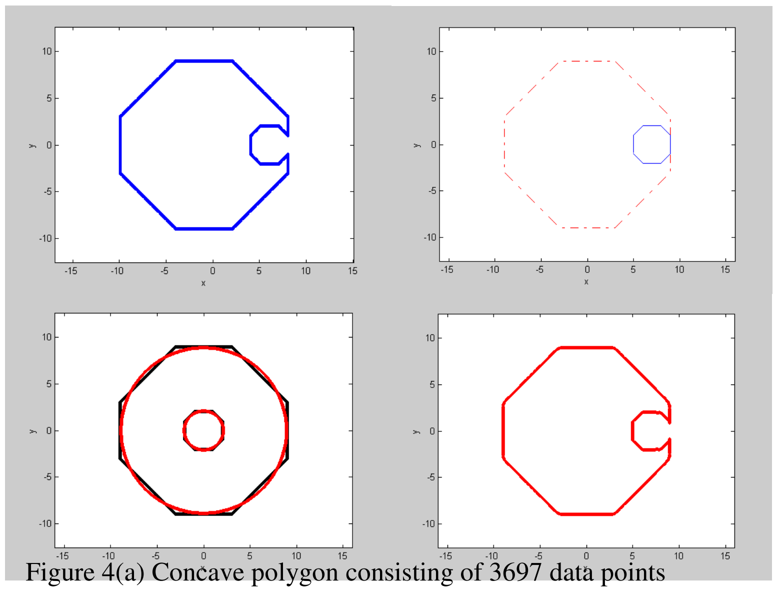


Figure 4(a) Concave polygon consisting of 3697 data points
Figure 4(b) Original shape decomposed into convex subsets
Figure 4(c) Convex subsets mapped to circles
Figure 4(d) Reconstruction of the original shape through inverse mapping process, using 20 derived coefficients, resulting in a compression ratio of 123:1.

References

[1] Raphael C. Gonzales and Richard E. Woods, *Digital Image Processing*, pp 307-394, Addison Wesley, MA.

[2] Yoram Gdalyahu and Daphna Weinshall, "Flexible Syntactic Matching of Curves and Its Application to Automatic Hierarchial Classification of Silhouettes," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, pp. 1312-1328, 1999.

[3] Dalila Megherbi, "A New Fast, Robust and Invariant Technique for 2-D Polygonal and Non-Polygonal Object Representation and Recognition based on Complex Variables and Conformal Mapping Methodologies" , Submitted to the IEEE journal Transaction on Pattern Recognition and Machine Intelligence.

[4] Dalila Megherbi, "How to map Arbitrary Shapes into Circles for Object Representation and Recognition, IMSE Technical Report #1, September 1998.

[5] Esther M. Arkin, et al, "An Efficient Computable Method for Comparing Polygonal Shapes," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol.13, pp. 209-216, 1991.

[6] Michael Shneier and Mohamed Abdel-Mottaleb, "Exploring the JPEG Compression Scheme for Image Retrieval," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, pp. 849-853, 1996.

[7] R.W. Picard and A.P. Pentland, "Introduction to the Special Section on Digital Libraries," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, pp. 769-770, 1996.

[8] Nalini K. Ratha, Kalle Karu, Shaoyun Chen and Anil K. Jain, "An Invariant Representation of an Object for Large Fingerprint Databases," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, pp. 779, Van Nostrand Reinhold, 1996.

[9] Ian H. Witten, Alistar Moffat and Timothy C. Bell, "Managing Gigabytes, Compressing and Indexing Documents and Images," Van Nostradn Reinhold, 1993.

[10] William B. Pennebaker and Joan L. Mitchell, "JPEG Still Image Compression Standard," Van Nostrand Reinhold, New York, 1993.

[11] Enrico Piazza, "Remote Sensed Multispectral Image Compression", part of the SPIE Conference on Mathematics of Data/Image Coding, Compression, and Encryption, San Diego, CA, July 1998. SPIE Vol. 3456.

[12] Michael F. Barnsley, "Fractal-Based Compression" AK Peters, Ltd., Wellesly, MA, 1993.

[13] Sadayasu Ono, Naohisa Ohta and Tomonori Aoyama "Super-High-Definition Images-Beyond HDTV", Artech House, Norwood, MA, 1995.

[14] Majid Rabbani and Paul W. Jones, "Digital Image Compression Techniques", SPIE Optical Engineering Press, Bellingham, WA, 1991.

[15] B. G. Lee, "A New Algorithm to Compute the Discrete Cosine Transform," IEEE Traans. Acous., Speech, Signal Proc., ASSP-32(6), 1243-1245 (1984).

[16] M. Vetterli and H. Nussbaumer, "Simple FFT and DCT Algorithms with Reduced Number of Operations," Signal Processing, 6, 267-278 (1984).

[17] Michal Etizion and Ari Rappoport, "On Compatible Star Decompositions of Simple Polygons," IEEE Transactions on Visualization and Computer Graphics, Jan – Mar 1997, Vol. 3, #1.

[18] M. Keil, "Decomposing a Polygon into Simpler Components," SIAM J. Computing, Vol 14, pp. 799-817, 1985.

[19] http://www.faqs/jpeg "How well does JPEG Compress Images?"