Shape Representation and Classification Using the Poisson Equation

Lena Gorelick[1,3] Meirav Galun[1,3] Eitan Sharon[2,3] Ronen Basri[1,3] Achi Brandt[1,3]

[1]Dept. of Computer Science and Applied Math.
The Weizmann Inst. of Science
Rehovot 76100, Israel

[2]Division of Applied Math.
Brown University
Providence, RI 02912

Abstract

Silhouettes contain rich information about the shape of objects that can be used for recognition and classification. We present a novel approach that allows us to reliably compute many useful properties of a silhouette. Our approach assigns for every internal point of the silhouette a value reflecting the mean time required for a random walk beginning at the point to hit the boundaries. This function can be computed by solving Poisson's equation, with the silhouette contours providing boundary conditions. We show how this function can be used to reliably extract various shape properties including part structure and rough skeleton, local orientation and aspect ratio of different parts, and convex and concave sections of the boundaries. In addition to this we discuss properties of the solution and show how to efficiently compute this solution using multigrid algorithms. We demonstrate the utility of the extracted properties by using them for shape classification.

1. Introduction

Silhouette contours contain detailed information about the shape of objects. In many cases it is possible given a silhouette to determine the parts that compose a shape, identify their local orientation and rough aspect ratio, and detect convex and concave sections of the boundaries. When a silhouette is sufficiently detailed people can readily identify the object, or judge its similarity to other shapes (see examples in Fig. 1).

Computer vision systems may use similar information to classify objects. Silhouettes may be available to these systems as a result of segmentation. Simple thresholding can be applied in applications involving fairly isolated objects, such as objects placed on a conveyer belt or characters in a document. More sophisticated algorithms may be used, with certain degree of success, to segment objects in cluttered scenes. In either case, properties of silhouettes extracted automatically and reliably provide (possibly in conjunction with additional properties such as color and texture) a powerful cue for recognition.

Figure 1: A collection of silhouettes.

In this paper we present a novel approach that allows us to reliably compute many useful properties of a silhouette. We consider a silhouette surrounded by a simple, closed contour. Based on the notion of random walks, we compute a function that assigns, for every internal point in the silhouette, a value reflecting the mean time required for a random walk beginning at the point to hit the boundaries. This function can be formalized as a partial differential equation, called the *Poisson equation*, with the silhouette contours providing boundary conditions. We then show how we can use the solution to the Poisson equation to reliably extract various properties of a shape including its part structure and rough skeleton, local orientation and aspect ratio of different parts, and convex and concave sections of the boundaries. In addition to this we discuss properties of the solution and show how to efficiently compute this solution using multigrid algorithms. We demonstrate the utility of the extracted properties by using them for shape classification.

The computer vision literature contains numerus examples for the use of properties extracted from silhouettes. Various studies utilize parts and skeleton structures to determine shape category [4, 20, 17]. Local properties are used for registration and recognition, as well as similarity judgement [1, 9, 21]. In addition, various categorization methods use "qualitative descriptions" of shape boundaries [3, 6]. Random walks are used in various vision

[3]Research was supported in part by the European Commission Project IST-2002-506766 Aim Shape and by the Binational Science foundation, Grant No. 2002/254. Ronen Basri was supported in part by the European Commission Project IST-2000-26001 VIBES. Achi Brandt was supported by the Israel Science Foundation grant No. 295/01, GIF 3982 and by the Israel Institute of Technology. The vision group at the Weizmann Inst. is supported in part by the Moross Foundation. The authors thank Yaara Goldschmidt for providing software for early experiments.

applications including perceptual grouping and segmentation [2, 10, 18, 24]. Finally, the Poisson equation is used in various applications such as optical flow and shape from shading [13, 16, 22].

The paper is divided as follows. Section 2 introduces the Poisson equation and its properties. Efficient multigrid solutions are discussed in Section 3. Section 4 describes how the solution can be used to extract various properties of a silhouette. Finally, Section 5 demonstrates the utility of these properties through some applications.

2. The Poisson Equation

Consider a silhouette S surrounded by a simple, closed contour. A sensible approach to inferring properties of the silhouette is to assign to every internal point a value that depends on the relative position of that point within the silhouette. One popular example is the distance transform, which assigns to every point within the silhouette a value reflecting its minimal distance to the boundary contour, and which can be computed by solving the Eikonal equation ($\|\nabla u\|^2 = 1$). An alternative approach is to place a set of particles at the point and let them move in a random walk until they hit the contour. Then we can measure various statistics of this random walk, such as the mean time required for a particle to hit the boundaries. This particular measure can be computed by solving a Poisson equation of the form:

$$\Delta U(x, y) = -1, \tag{1}$$

with $(x, y) \in S$, where the Laplacian of U is defined as $\Delta U = U_{xx} + U_{yy}$, subject to the boundary conditions $U(x, y) = 0$ at the bounding contour ∂S. The connection to random walk is evident by noticing that, according to the Poisson equation, the value at every point is a constant plus the average value of its neighbors.

Fig. 2 shows the solution to the Poisson equation obtained for the silhouettes in Fig. 1. The level sets of U represent smoother versions of the bounding contour with the external protrusions (the limbs, head, and tail) disappearing already at relatively low values of U. This is different from the distance transform, which smoothes the shape near concavities while introducing discontinuities near convex sections of the contour. Also unlike the distance transform in which every value is determined by a single contour point (the nearest), the values assigned by the Poisson equation take into account many points on the boundaries and so they reflect more global properties of the silhouette. Below we exploit these properties of the Poisson solution to characterize a silhouette using measures constructed with derivatives of the solution.

Poisson's equation arises in gravitation and electrostatics. Here we only mention a few of its relevant properties. The solution to the Poisson equation exists and is unique for any closed region with boundary conditions given by any integrable function. Uniqueness is shown by noticing that the solution to the related homogeneous equation $\Delta U^h = 0$ (called the *Laplace equation*) with zero boundary conditions is identically zero. Moreover, the values of U along any closed curve within S determine the values of U inside the region bounded by this curve, but they are insufficient to uniquely determine the values of U outside the curve.

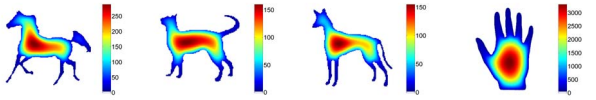

Figure 2: Solutions to the Poisson equation for the silhouettes in Fig. 1.

Figure 3: Noisy boundary (left) and two level sets of low values (right).

For silhouettes described by conics the Poisson equation takes a particularly simple form. Consider a silhouette composed of the points (x, y) satisfying

$$P(x, y) = ax^2 + by^2 + cxy + dx + ey + f \leq 0. \tag{2}$$

Then, it can be readily verified that the solution is given by

$$U(x, y) = -\frac{P(x, y)}{2(a + b)}. \tag{3}$$

In this case the level sets of U simply contain a nested collection of scaled versions of the boundaries, where the value of U increases quadratically as we approach the center.

For silhouettes described by more complicated equations the level sets of U represent smoother versions of the bounding curve (as is shown in Fig. 3). To see this, consider a line on which U oscillates (e.g., as a result of a ragged bounding contour). As we proceed toward the inside of the region the oscillation will attenuate at an exponential rate. We can illustrate this "interior regularity" property (common to all smooth elliptic equations [7, 5]) by solving (1) within the infinite band $0 \leq y \leq L$ with boundary conditions $U(x, 0) = e^{ix/\lambda}$ and $U(x, L) = 0$. The solution is given by $U = e^{-y/\lambda}e^{ix/\lambda} + 0.5y(L-y)$. Consider the value of U along a line parallel to the x-axis (constant y). The left term oscillates with the same wavelength as on the lower boundary, but with a decaying amplitude ($e^{-y/\lambda}$), whereas the right term is constant. The faster the oscillation (smaller λ) the faster the decay of its amplitude. Note that due to the linearity of the Poisson equation this analysis also applies to oscillations expressed as superpositions of frequencies.

3. Multigrid Solutions

Numerical solutions to the Poisson Equation can be obtained using successive Jacobi or Gauss-Seidel relaxations, but their convergence rate is slow. It requires $O(n^2)$ computer operations to reach an adequate solution for a silhouette region with n pixels. This slowness is a result of the fact that after several relaxation sweeps the obtained approximation has a smooth error, which can only be slightly reduced by each relaxation sweep. Such a smooth error can however be approximated on a coarser grid. This in fact is what multigrid algorithms do.

A multigrid solver consists of several relaxation passes, followed by averaging the residual equations (the equations for the current, smooth error function), to represent them on a coarser grid (where the distance between neighboring grid points is twice the fine-grid distance, i.e., twice the pixel size). The approximate solution to the resulting coarse grid equations is interpolated to the fine grid, yielding a good approximation for the smooth error, which can thus be used to correct the previous, erroneous solution. This combination of several relaxation sweeps followed by a correction from a coarser grid is called *a multigrid cycle*. One such cycle typically reduces the error by an order of magnitude. The coarse equations in this cycle are themselves approximately solved by a similar cycle, i.e., several relaxation sweeps (on the coarse-grid equations) followed by a correction from *a still coarser* grid (with meshsize four times the pixel size); and so recursively, until, at a very coarse grid, the equations are solved directly. (For recent multigrid textbooks see [23]).

Since the computational work at the increasingly coarser levels diminishes geometrically, the overall cost of a multigrid cycle is only a fraction more than the cost of the several fine-grid relaxation sweeps. Several such cycles are enough to produce a very accurate solution, so the total cost of the multigrid solver is just $O(n)$. In fact, for all the applications described below, which largely require qualitative features rather than precise numerical values, only a crude accuracy is needed. This can be achieved by applying just one, simplified cycle, which employs very naive boundary conditions at the coarse levels (placing the boundary at the nearest coarse grid points, instead of modifying the nearby coarse equations to account for the fine, pixel-level location of the boundary). We observed no qualitative difference in the measured features between this one-cycle solver and a fully accurate, several-cycle solver. This plain solver costs less than two dozens operations per silhouette pixel.

The interior regularity property described in Sec. 2 implies that at a distance r from the boundary a grid with meshsize $h = O(r)$ suffices to preserve the discretization error. This property can be utilized to further improve the complexity of the multigrid algorithm above. The modified version will exploit non-uniform grids with meshsize dependent on the distance r from the boundary. The complexity of such a solver is linear in the number of contour points, while its accuracy is unchanged.

Finally, the solution obtained may be noisy near the boundaries due to discretization. To reduce this noise we may apply as a post-processing stage a few relaxation sweeps enforcing $U_{xx} + U_{yy} = -1$ inside the silhouette and $U_{xx} + U_{yy} = 0$ outside the silhouette. This will smooth U near the boundaries and hardly affect more inner points.

4. Extracting Silhouette Properties

We can use the Poisson equation to extract a wide variety of useful properties of a silhouette. In this section we provide a few examples. We begin by showing how the Poisson equation can be used to segment a silhouette into parts. Next, we show how we can identify corners at various resolution scales and derive a skeleton structure. Finally, we show how we can locally judge the orientation and rough aspect ratio of portions of the silhouette.

4.1. Hierarchical shape representation

Complex shapes can often be naturally described as a collection of parts. In particular, many common objects can undergo articulated motion in which every part moves rigidly while stretching or rotating with respect to the other parts of the object. For these reasons a number of recognition schemes were proposed that rely on part representations (e.g., [4, 20, 17]). In this section we show how we can use the Poisson equation to divide a shape into parts.

We will focus on shapes composed of a central part (e.g, the torso of an animal) to which exterior parts are connected (e.g., the head, limbs, and tail). In such a case the highest values of U are obtained within the central part. Using an appropriate threshold we can identify the central part (see Figure 4). However, in this case we obtain a smooth version of the central part and lose the portions of the central part that are near the boundaries since those portions yield low values of U.

We can include the portions of the central part that fall near the boundaries by noticing that these portions must have a high gradient. Define

$$\Phi = U + {U_x}^2 + {U_y}^2. \tag{4}$$

Φ has several interesting properties. First it is constant on a disc ($\Phi = r^2/4$ on a disc of radius r). On other shapes it has a saddle point exactly where U is maximum. For example, on an ellipse we obtain a saddle point at the center with low values along the major axis and high values along the minor axis. In more complex shapes there may be additional saddle points at minor parts, indicating the hierarchy of the parts. The highest values of Φ are obtained near concavities. This follows the fact that U grows faster with distance

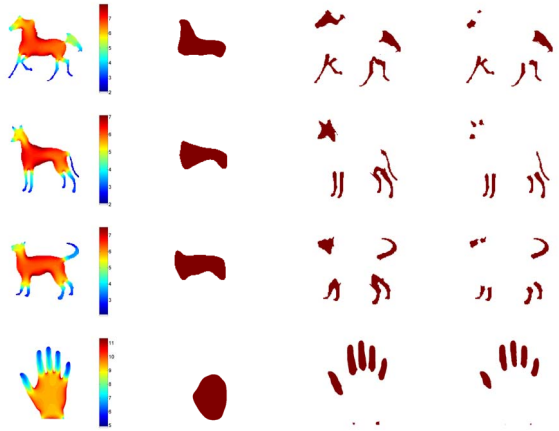

Figure 4: From left to right: $\log(\Phi)$ computed for several silhouettes, central blobs extracted using 55 highest percentiles of U, and exterior parts extracted using uniform thresholds (78 and 66 low percentiles of $\log \Phi$).

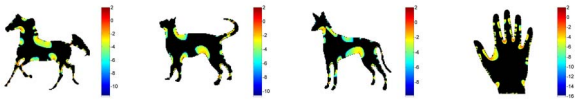

Figure 5: Detecting concavities in different shapes using Ψ. The figure highlights $\log|\Psi|$ for all silhouette points with negative Ψ.

from concavities than from convexities, and so the gradient magnitude is higher.

By simply thresholding Φ we can divide a shape into parts. Fig. 4 shows a few examples. Note that by changing the threshold we can extract also very small parts, such as the ear or a horseshoe.

4.2. Identifying corners

For every point inside the silhouette we can evaluate the curvature of the level set passing through this point using

$$\Psi = -\nabla \cdot \left(\frac{\nabla U}{\|\nabla U\|} \right) \tag{5}$$

[19]. High level values of Ψ mark locations where level sets are significantly curved. Since the level sets of U are smooth versions of the bounding contour this measure can be used, for example, to detect corners at different scales.

Negative values of Ψ identify all shape concavities, as is shown in Fig. 5. High negative values are obtained near local, sharp concavities (e.g., the horse's neck), and somewhat lower values in an extended area are obtained for large-scale concavities (e.g., the horseback). High positive values of Ψ will reveal all convexities, along with ridge (skeleton-like) locations. It is therefore preferable to detect convex corners using low values of Φ, see Fig. 6.

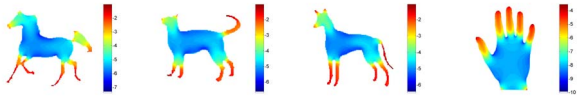

Figure 6: Detecting convex sections in different shapes using low values of Φ. The figure shows $\log(1/\Phi)$ for all silhouette points. Notice that convex points are highlighted.

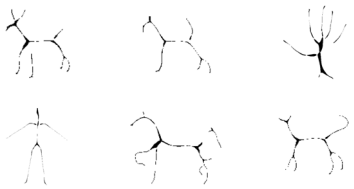

Figure 7: Rough skeletons computed with $\tilde{\Psi}$. To obtain narrow skeletons we used a threshold, and repeated the computation of $\tilde{\Psi}$ on the obtained region.

4.3. Skeleton

To further emphasize the skeleton of a shape we may use a scale-invariant version of Ψ defined as:

$$\tilde{\Psi} = -\frac{U \cdot \Psi}{\|\nabla U\|}. \tag{6}$$

This measure emphasizes the skeletal structure, while attenuating the response for corners near the boundaries since near the boundaries U is small, while near ridge location $\|\nabla U\|$ is small. The skeleton obtained using the Poisson equation is similar, but not identical to the skeleton obtained with the brushfire algorithm. Fig. 7 shows rough skeletons computed with $\tilde{\Psi}$.

4.4. Local orientation and aspect ratio

The Poisson equation can be used also to estimate the local orientation and dimension of a shape. One example in which this may be useful is character recognition (OCR). Characters often can be described as a collection of elongated, narrow strokes, most often with no noticeable central part. The lack of a central part makes the method discussed in Section 4.1 above inappropriate for characters. We thus need a different procedure that can identify the parts that compose a character. Furthermore, it is important that this method will be applicable for very thin (pixel-wide) as well as thick characters.

A natural way to divide a character into parts is to identify the local orientation of the part and divide the character into sections of different orientation. This can be done simply by looking at the second derivatives of U. In a narrow elongated section of a shape the level sets of U are largely parallel to the orientation of that section. The second derivative of U along a level set is generally very small, while the

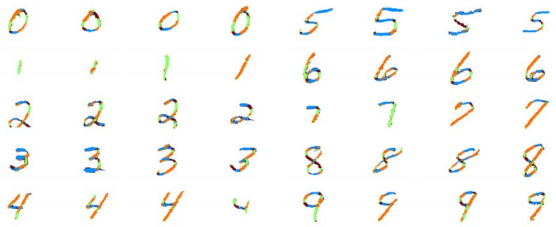

Figure 8: A schematic representation of the second derivatives for hand-written numerals: each pixel is assigned a color according to the direction in which the derivative at this pixel is minimal (indicating that the pixel lies in a part that is elongated in this direction), i.e. green for the vertical direction, blue for horizontal, brown and orange for the two diagonals.

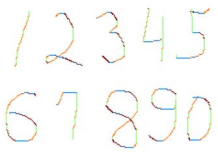

Figure 9: A schematic representation of the second derivatives (similar to the previous figure) for pixel-wide numerals.

second derivative at the perpendicular direction must complement it to -1, and so it must be much larger (in absolute value). We may therefore identify the local orientation at each pixel by detecting the orientation θ in which $U_{\theta\theta}$ is small.

Figures 8 and 9 show a schematic representation of local orientation for sets of thick and thin hand written numerals. In the case of pixel-wide characters we may use the smoothing post-processing described in Section 3, although the detection of orientation seems to work well also without this smoothing. Note that the detection of orientation can be applied everywhere within the shape, and so it is fairly insensitive to small perturbations of the contour. In Section 5 below we show an example of using such detection in a numeral recognition task. One can also build on this orientation measure other useful processes and recognition tools. In particular, we may segment each numeral to its constituent strokes by identifying sections in which orientation changes smoothly. Each stroke can then be characterized by various properties, such as its location and several moments of its orientation as function of arclength. These ideas can also be applied to identifying directions of skeleton parts.

We can further generalize the analysis above to arbitrary shapes by constructing measures that estimate locally the second order moments of a shape near any given point. Consider a shape defined by a second order polynomial $P(x,y) < 0$, as in (2). As we mentioned in Section 2, every level set of U is simply a scaled version of $P(x,y)$ (3). If we now consider the Hessian matrix of U we obtain

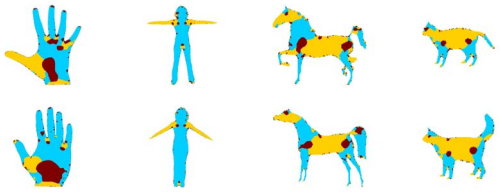

Figure 10: A schematic representation of orientations computed via the Hessian matrix for several shapes. Turquoise color represents vertical regions ($|\alpha| \geq \frac{\pi}{4}$), yellow for horizontal ($|\alpha| < \frac{\pi}{4}$), and brown for isotropic sections $\sqrt{|\lambda_{\min}/\lambda_{\max}|} \geq \frac{2}{3}$). It can be seen that pairs of similar shapes give rise to similar orientation measures, but notice the thumb (left figures), which changes its orientation from near horizontal (top) to near vertical (bottom).

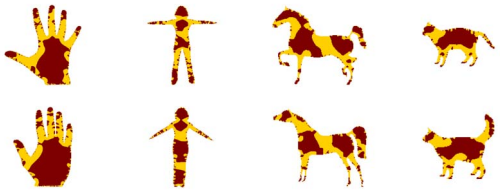

Figure 11: Elliptic (brown) and hyperbolic (yellow) sections computed with the Hessian matrix for several shapes (sign of $\lambda_{\min} \cdot \lambda_{\max}$).

at any given point exactly the same matrix, namely

$$H(x,y) = -\frac{1}{a+b}\begin{pmatrix} a & c/2 \\ c/2 & b \end{pmatrix}. \qquad (7)$$

This matrix is in fact the second moment matrix of the entire shape, scaled by a constant. The eigenvectors and eigenvalues of H then will reveal the orientation of the shape, its aspect ratio, and will indicate whether the shape is elliptic or hyperbolic.

For more general shapes the Hessian will vary continuously from one point to the next, but we can treat it as providing a measure of moments of shape felt locally by the point. Figures 10 and 11 show a few examples. In these figures $\lambda_{\min}(x,y)$ and $\lambda_{\max}(x,y)$ denote the eigenvalues of the Hessian matrix s.t. $|\lambda_{\min}(x,y)| \leq |\lambda_{\max}(x,y)|$ and $\alpha(x,y)$ denotes the orientation of the leading eigenvector.

5. Applications

Below we describe preliminary classification experiments which demonstrate the utility of properties extracted with the Poisson equation. In these experiments, we represent a shape by a collection of features derived for the shape using the Poisson solution (1). The features we use are weighted moments of the form

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} w(x,y)x^p y^q dxdy, \qquad (8)$$

in which for every feature we substitute for $w(x,y)$ a different Poisson-based measure (described below). We then

use these features to train a classifier with multiple decision trees, each trained on a random subset of the training set. At each level of a tree we compute a multiclass Fisher discriminant [8] and find along the obtained direction the split that achieves maximal impurity decrease. We then test the algorithm by applying it to an unseen randomly chosen test set. We compare our results with those reported in the literature and with the results obtained using the same classifier when the features used include ordinary shape moments (computed as in (8) with $w(x,y) = \chi(x,y)$, the characteristic function of the shape).

Following is a description of the features used and the results obtained in each of the experiments.

5.1. Hand-Written Numerals

In the first set of experiments we learn to classify handwritten numerals. We used two types of measures as weights in (8). The first measure, L_θ, identifies local orientation by detecting the orientation in which the second derivative of the solution is near zero. This measure is defined by

$$L_\theta(x,y) = e^{-(\beta|\tilde{U}_{\theta\theta}(x,y)|)}, \tag{9}$$

where $\tilde{U}_{\theta\theta}(x,y)$ denotes a three-pixel average of $U_{\theta\theta}(x,y)$, the second derivative of U in one of the four directions $\theta \in \{0, \pi/2, \pi/4, -\pi/4\}$ (we used $\beta = 3$). The second measure, $J_{\theta_1\theta_2}(x,y)$, identifies junctions by taking the geometric mean of two detectors, $L_{\theta_1}(x,y)$ and $L_{\theta_2}(x,y)$ in the six pairs of orientations (θ_1, θ_2) s.t. $\theta_1, \theta_2 \in \{0, \pi/2, \pi/4, -\pi/4\}$ and $\theta_1 < \theta_2$. We used these moments (along with two global properties, the aspect ratio of the numeral and the ratio between the product of PCA length and width and the area of the numeral) to construct a vector of 210 features for each numeral.

We then used a random subset of 20000 numerals for training. The algorithm constructed 50 decision trees, each trained by a random collection of 3000 numerals from the training set. We then tested the algorithm by applying it to 1000 unseen randomly chosen numerals.

We first applied this procedure to numerals from the NIST special database 19 [12] (downsampled to fit in a 64×64 pixel box, centered about their centroid and brought to a uniform scale). The average error rate obtained in 10 random experiments was $1.08\% \pm 0.19\%$. (One of our errors was due to mislabeling.) For comparison, with ordinary shape moments of up to order 12 (including the two global features, yielding 90 features) we obtained an error rate of $3.7\% \pm 0.35\%$. Further experiments with moments up to all orders between 4 and 20 (resulting in 14-230 features) yielded worse performance. In addition, our results favorably compare to the $3.58\% \pm 0.13\%$ reported in [11].

We then applied the same procedure to numerals from the MNIST database [14]. The average error rate obtained in 10 random experiments was $1.5\% \pm 0.18\%$. For comparison, with shape moments of orders up to 11 (yielding 77 features) we obtained $5.4\% \pm 0.28\%$. Further experiments with moments up to any order between 4 and 20 yielded worse performance. Our results are superior to many of the algorithms tested in the benchmark [15], including RBF, Neural nets, PCA, and linear classifiers, but are inferior to the best results published for boosted LeNet-4 [15] (0.7%) and for shape context [3] (0.63%). Our algorithm, however, is significantly faster. For example, our non-optimized Matlab implementation requires less than 15 minutes of training with 20K numerals.

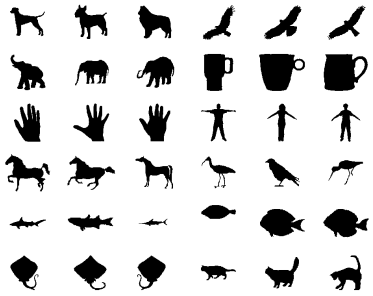

Figure 12: A collection of silhouettes from the database.

5.2. Natural Silhouettes

The next experiment demonstrates the utility of our method for general shapes. We collected a database of silhouettes of natural objects and expanded it with the most variable classes from [20]. The database contained 490 silhouettes of 12 classes (see Fig. 12). We centered the silhouettes about their centroid and brought them to a uniform scale, and then solved the Poisson equation (1). We characterized every silhouette using two measures. The first measure is analogous to L_θ (9). It identifies vertical and horizontal regions of a shape by detecting points for which the orientation computed with the Hessian matrix (Section 4.4) is close to either zero or $\pi/2$. This measure is defined by

$$H_\theta(x,y) = e^{-\gamma|\theta - |\alpha(x,y)||}, \tag{10}$$

where $\alpha(x,y)$ is the principal direction felt locally at (x,y) ($\frac{-\pi}{2} \leq \alpha(x,y) \leq \frac{\pi}{2}$) and γ is a constant (we used $\gamma = 3$). We evaluated H_θ at every point with ratio $\sqrt{|\lambda_{\min}/\lambda_{\max}|} \leq \frac{1}{2}$ for the two orientations $\theta \in \{0, \frac{\pi}{2}\}$. The second measure identifies concave regions as well as elongated convex sections by emphasizing points with high values of Φ (Section 4.1). Denote by $\hat{\Phi}(x,y)$ the function $\Phi(x,y)$ centered about its saddle point value (the value at the point where U is maximal) and normalized so that its maximal absolute value is 1. We define

$$K_\Phi(x,y) = \frac{1}{1 + e^{-\delta\hat{\Phi}(x,y)}} \tag{11}$$

(we used $\delta = 4$).

In the classification experiment we used a random selection of 350 of the silhouettes for training and the remaining 140 silhouettes for testing. We then computed moments up to order 3 as in (8) substituting for $w(x, y)$ the two vertical and horizontal measures H_θ and the measure K_Φ. We used in addition the direction of the principal axis, aspect ratio, and ratio of axial standard deviations, resulting in 33 features per silhouette. Using the same classification algorithm as with the numerals (constructing 10 trees, each trained with a random selection of 300 silhouettes), in a 100 random experiments, we obtained an average error rate of $3.8\% \pm 1\%$. For comparison, with shape moments of orders up to 6 (including the three additional features, resulting in 26 features) we obtained $6.9\% \pm 1.3\%$. Experiments with moments of different orders (we tried all orders between 3 and 7, resulting in 8-34 features) yielded worse performance.

6. Conclusion

Solutions to the Poisson equation provide rich descriptive information that can be used to compute useful properties of shape silhouettes. In this paper we derived several such properties, described how to compute them efficiently using multigrid solvers, and applied them in classification tasks. We are unaware of any other method that can produce a scalar field whose thresholding decomposes an object into parts. Moreover, existing scalar fields (e.g. the distance transform) do not yield such rich descriptions simply by differentiation. It is also worth noting that solutions to the Poisson equation, and the properties extracted from its solution can be applied with very little change also to objects in higher dimensions.

References

[1] Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9(7):1545–1588, 1997.

[2] J. August and S.W. Zucker. Sketches with curvature: The curve indicator random field and markov processes. *PAMI*, 25(4):387–400, 2003.

[3] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *PAMI*, 24(4):509–522, 2002.

[4] I. Biederman. Human image understanding: recent research and a theory. *CVGIP*, 32:29–73, 1985.

[5] A. Brandt. Interior estimates for second order elliptic differential (or finite-difference) equations via the maximum principle. *Israel J. Math.*, 7:95–121, 1969.

[6] S. Carlsson. Order structure, correspondence and shape based categories. *International Workshop on Shape, Contour and Grouping , Springer Lecture Notes in Computer Science*, page 1681, 1999.

[7] A. Douglis and L. Nirenberg. Interior estimates for elliptic system of p.d.e. *Comm. Pure Appl. Math.*, 8:503–538, 1955.

[8] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New Uork, 1 edition, 1973.

[9] Y. Gdalyahu and D. Weinshall. Flexible syntactic matching of curves and its application to automatic hierarchical classification of silhouettes. *PAMI*, 21(12):1999, 1999.

[10] D. Geiger, K. Kumaran, and L. Parida. A computational view of visual organization for figure/ground separation. *CVPR*, pages 155–160, 1996.

[11] M. Giudici, F. Queirolo, and M. Valle. Evaluation of gradient descent learning algorithms with adaptive and local rate for hand-written numerals. *ESANN*, pages 289–294, 2002.

[12] P.J. Grother. Nist special database 19, handprinted forms and characters database. *Visual Image Processing Group, Advanced Systems Division, National Institute of Standards and Technology, http://www.nist.gov/srd/nistsd19.htm*, 1995.

[13] S.H. Lai and B.C. Vemuri. An o(n) iterative solution to the poisson equation in low-level vision problems. *CVPR*, pages 9–14, 1994.

[14] Y. LeCun. The mnist database of handwritten digits. *NEC Research Institute, http://yann.lecun.com/exdb/mnist/index.html*.

[15] Y. LeCun, L. Botou, L. Jackel, H. Drucker, C. Cortes, J. Denker, I. Guyon, U. Muller, E. Sackinger, P. Simard, and V. Vapnik. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural Networks*, pages 261–276, 1995.

[16] J. Li and A. Hero. A spectral method for solving elliptic equations for surface reconstruction and 3d active contours. *ICIP*, III:1067–1070, 2001.

[17] D. Marr and H.K. Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *Proc. of the Royal Society, London*, B200:269–294, 1978.

[18] D. Mumford. Elastica and computer vision. *Algebraic Geometry and Its Applications, Chandrajit Bajaj (ed.), Springer Verlag*, pages 491–506, 1994.

[19] S. Osher and N. Paragios, editors. *Geometric Level Set Methods in Imaging, Vision, and Graphics*. Springer, New York, 2003.

[20] T.B. Sebastian, Philip N. Klein, and Benjamin B. Kimia. Shock-based indexing into large shape databases. *ECCV (3)*, pages 731–746, 2002.

[21] T.B. Sebastian, P. Klien, and B.B. Kimia. On aligning curves. *PAMI*, 25(1):116–125, 2003.

[22] T. Simchony, R. Chellappa, and M. Shao. Direct analytical methods for solving poisson equations in computer vision problems. *PAMI*, 12(5):435–446, May 1990.

[23] U. Trottenberg, C. Oosterlee, and A. Schuller. *Multigrid*. Academic Press, 2001.

[24] L.R. Williams and D.W. Jacobs. Stochastic completion fields: a neural model of illusory contour shape and salience. *Neural Computation*, 9(4):837–858, 1997.