Neurocomputing 399 (2020) 399–413

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Robust ellipse fitting based on Lagrange programming neural network and locally competitive algorithm

Zhanglei Shi[a], Hao Wang[a], Chi-Sing Leung[a,*], Hing Cheung So[a], Junli Liang[b], Kim-Fung Tsang[a], Anthony G. Constantinides[c]

[a] City University of Hong Kong, Hong Kong
[b] Northwestern Polytechnical University, Hong Kong
[c] Imperial College, Hong Kong

ARTICLE INFO

Article history:
Received 25 April 2019
Revised 14 February 2020
Accepted 23 February 2020
Available online 26 February 2020

Communicated by Dr. Yiu-ming Cheung

Keywords:
Ellipse fitting
Outlier
Real-time solution
Lagrange programming neural network (LPNN)
Locally competitive algorithm (LCA)

ABSTRACT

Given a set of 2-dimensional (2D) scattering points, obtained from the edge detection process, the aim of ellipse fitting is to construct an elliptic equation that best fits the scattering points. However, the 2D scattering points may contain some outliers. To address this issue, we devise a robust ellipse fitting approach based on two analog neural network models, Lagrange programming neural network (LPNN) and locally competitive algorithm (LCA). We formulate the fitting task as a nonsmooth constrained optimization problem, in which the objective function is an approximated l_0-norm term. As the LPNN model cannot handle non-differentiable functions, we utilize the internal state concept of LCA to avoid the computation of the derivative at non-differentiable points. Simulation results show that the proposed ellipse fitting approach is superior to several state-of-the-art algorithms.

1. Introduction

Fitting of geometric primitives with given 2-dimensional (2D) scattering points is required in many research areas, such as physics [1], biology [2,3], gait periodicity detection [4], healthcare technology [5], and computer vision [6]. In particular, an ellipse is a common geometric primitive in image processing. Generally, ellipse fitting is more difficult than circle fitting because the equation of ellipse is more complicated than that of circle.

Numerous ellipse fitting algorithms have been developed in the literatures. They can be roughly classified into two categories. The first category involves Hough transform (HT) and its variants [7,8]. Its basic idea is to search the five parameters of the ellipse in a 5-dimensional (5D) space. Apparently, the searching process is computationally costly. The second one is based on the least squares (LS) methodology. Its key idea is to calculate the elliptical parameters by minimizing an error metric between the geometric primitives and the scattering points [9]. In general, the LS category is more computationally efficient than the first category. The LS category can be further divided into two sub-categories, geometric and algebraic.

In the geometric sub-category, the error metric is the sum of the orthogonal distances between the 2-D scattering points and the constructed ellipse [10,11]. In the algebraic sub-category [12–15], for each scattering point, the fitting score is based on the algebraic distance. The algebraic based algorithms were extensively studied because they are generally simple and computationally attractive. Among various algebraic based algorithms, the constrained least squares (CLS) [15] is a representative algorithm, which introduces a unit-norm constraint on the elliptical parameter vector. Although the algebraic based algorithms work very well in many cases, they are quite sensitive to outliers. It should be noticed that the 2D scattering points are usually acquired from the edge detection process. Hence it is difficult to avoid disturbances including outliers. So there is a need to devise robust algebraic based algorithms. Recently, a few robust ellipse fitting numerical algorithms have been proposed, including the sparsity based method (SBM) [16] and the robust CLS (RCLS) [17]. The former utilizes the l_1-norm to resist outliers and calculates the elliptical parameters by solving a second-order cone programming problem. The latter introduces the maximum correntropy criterion and quadratic constraint to enhance robustness.

Bio-inspired techniques, such as artificial neural networks [18,19], evolutionary [20], and genetic algorithms [21,22], have been used for engineering applications. In particular, analog

* Corresponding author.
E-mail address: eeleungc@cityu.edu.hk (C.-S. Leung).

https://doi.org/10.1016/j.neucom.2020.02.100
0925-2312/

neural circuits for solving constrained optimization problems have been investigated over twenty years [18,19,23–27]. When we require realtime/interactive solutions [18,19,28], the analog neural circuit approach is more preferable.

In the analog neural circuit approach, we use a number of neurons to hold the decision variables of the optimization problem and develop the neural dynamics to guide the neuron state transition. The solution of the optimization problem is obtained by measuring the neuron state at the equilibrium state of the network. Tank and Hopfield [18] demonstrated that the simple Hopfield network model is able to solve various kinds of optimization problems, such as analog-to-digital conversion (ADC).

In [19,29–31] a number of models were proposed to solve various nonlinear constrained optimization problems. Also, various projection neural network models [24,25,30,32] were proposed in the last two decades. However, many existing models are designed for solving a dedicated constrained optimization problem. For instance, in [27], the model was designed for the quadratic programming problem with the box constraint.

The Lagrange programming neural network (LPNN) approach [33–37] provides a general framework for solving various constrained optimization problems. With the augmented term concept, the LPNN approach is able to solve nonconvex optimization problems. Recently, some new applications of using LPNN approach were reported [36–39], including sparse approximation, target localization, and waveform design for radar systems. However, the original LPNN framework is applicable to the differentiable objective function and constraints only.

This paper develops a robust ellipse fitting approach based on the LPNN approach [33–37,40]. The l_p-norm ($p \leq 1$) based objective function [16] is able to achieve robustness against outlier samples. Especially, in terms of suppressing the effect of outlier samples, the l_0-norm based objective function is much better.

This paper exploits the LPNN formulation for ellipse fitting with the l_0-norm based objective function. We call the proposed approach $\boldsymbol{l_0}$**-LPNN**. Since the traditional LPNN framework requires that its objective function and constraints should be twice differentiable, we adopt the locally competitive algorithm (LCA) [41,42] to avoid the computation of derivatives at non-differentiable points by utilizing the hidden state concept. Simulation results show that the proposed ellipse fitting approach is superior to several state-of-the-art algorithms.

The rest of this paper is organized as follows. The backgrounds of ellipse fitting, the LPNN and LCA models are described in Section 2. In Section 3, the proposed ellipse fitting algorithm is developed. The local stability of the LPNN approach is proved in Section 4. Numerical results for algorithm evaluation and comparison are provided in Section 5. Finally, conclusions are drawn in Section 6.

2. Background

2.1. Notation

We use a lower-case or upper-case letter to represent a scalar while vectors and matrices are denoted by bold lower-case and upper-case letters, respectively. The transpose operator is denoted as $(\cdot)^{\mathrm{T}}$, and $\boldsymbol{I}$ and $\boldsymbol{0}$ represent the identity matrix and zero matrix of appropriate dimensions, respectively. Other mathematical symbols are defined in their first appearance.

2.2. Ellipse fitting

An axis-aligned ellipse, centered at (c_x, c_y), can be expressed as:

$$\frac{(x-c_x)^2}{a^2}+\frac{(y-c_y)^2}{b^2}=1. \tag{1}$$

where a and b are the radii along the two axes, respectively. This particular parametric model is frequently used in the diameter control system of silicon single crystal growth [43]. For the more general case, a non-axis aligned ellipse centered at (c_x, c_y) with a counter-clockwise rotation of θ can be described as

$$\frac{((x-c_x)\cos\theta+(y-c_y)\sin\theta)^2}{a^2}+\frac{(-(x-c_x)\sin\theta+(y-c_y)\cos\theta)^2}{b^2}=1. \tag{2}$$

The task of ellipse fitting is to find the five parameters $\{a, b, c_x, c_y, \theta\}$. However, it is very difficult to estimate them directly because Eq. (2) is highly nonlinear. Instead, many ellipse fitting algorithms [44–46] consider the second-order polynomial model, given by

$$Ax^2+Bxy+Cy^2+Dx+Ey+F=0, \tag{3}$$

where the six parameters $\{A, B, C, D, E, F\}$ are related to $\{a, b, c_x, c_y, \theta\}$ as:

$$A=\frac{\cos^2\theta}{a^2}+\frac{\sin^2\theta}{b^2}, \tag{4}$$

$$B=2\cos\theta\sin\theta\left(\frac{1}{a^2}-\frac{1}{b^2}\right), \tag{5}$$

$$C=\frac{\sin^2\theta}{a^2}+\frac{\cos^2\theta}{b^2}, \tag{6}$$

$$D=\frac{-2c_x\cos^2\theta-2c_y\sin\theta\cos\theta}{a^2}+\frac{-2c_x\sin^2\theta+2c_y\sin\theta\cos\theta}{b^2}, \tag{7}$$

$$E=\frac{-2c_y\sin^2\theta-2c_x\sin\theta\cos\theta}{a^2}+\frac{-2c_y\cos^2\theta+2c_x\sin\theta\cos\theta}{b^2}, \tag{8}$$

$$F=\frac{(c_x\cos\theta+c_y\sin\theta)^2}{a^2}+\frac{(c_x\sin\theta-c_y\cos\theta)^2}{b^2}-1. \tag{9}$$

Let $\mathcal{D}=\{(x_i, y_i): i=1,\ldots,N\}$ be a set of 2-D scattering points of an ellipse. Denote

$$\boldsymbol{\alpha}=[A, B, C, D, E, F]^{\mathrm{T}}, \tag{10}$$

$$\boldsymbol{x}_i=[x_i^2, x_iy_i, y_i^2, x_i, y_i, 1]^{\mathrm{T}}, \forall i, \tag{11}$$

$$\boldsymbol{X}=[\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N]. \tag{12}$$

In the absence of measurement errors, from (3), we have

$$\boldsymbol{X}^{\mathrm{T}}\boldsymbol{\alpha}=[\boldsymbol{x}_1^{\mathrm{T}}\boldsymbol{\alpha},\ldots,\boldsymbol{x}_N^{\mathrm{T}}\boldsymbol{\alpha}]^{\mathrm{T}}=\boldsymbol{0}. \tag{13}$$

In a noisy environment, for a scattering point (x_i, y_i), i.e., $(\boldsymbol{x}_i=[x_i^2, x_iy_i, y_i^2, x_i, y_i, 1]^{\mathrm{T}})$, we have

$$\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\alpha}\neq 0. \tag{14}$$

The absolute value $|\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\alpha}|$ is called the "algebraic distance", which can be used to measure the fitting error of a point (x_i, y_i) [46].

The traditional CLS algorithm considers the following constrained optimization problem:

$$\min_{\boldsymbol{\alpha}} \left\|\boldsymbol{X}^{\mathrm{T}}\boldsymbol{\alpha}\right\|_2^2 \tag{15a}$$

$$\text{s.t. } \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\alpha} = 1. \tag{15b}$$

The objective function in (15 a) is the sum of squared algebraic distances. In (15 b), the unit-norm constraint is used to avoid the redundant solutions (the solutions with linear correlation) and the trivial solution ($\boldsymbol{\alpha} = \mathbf{0}$). The CLS approach is efficient for ellipse fitting, provided that the noise in data obeys Gaussian distribution. When the data set contains impulsive disturbances or even outliers, the CLS solution may have a large deviation from the actual ellipse.

It is worth pointing out that the CLS solution may also correspond to a hyperbola or parabola [15] because these two geometric primitives can be expressed by (13) as well. To eliminate these possibilities, an additional constraint is introduced, given by

$$B^2 - 4AC < 0, \tag{16}$$

which aims at constraining the solution to be an ellipse [17].

2.3. Lagrange programming neural network

The LPNN is an analog neural network computational approach. It can be used to solve a general nonlinear constrained optimization problem [33], given by

$$\min_{\boldsymbol{z}} \ f(\boldsymbol{z}) \tag{17a}$$

$$\text{s.t. } \boldsymbol{h}(\boldsymbol{z}) = 0, \tag{17b}$$

where $\boldsymbol{z} = [z_1, \ldots, z_n]^{\mathrm{T}}$ is the variable vector being optimized, $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function, $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^m$ with $m < n$ represents m equality constraints, and f and $\boldsymbol{h}$ should be twice differentiable.

The first step in the LPNN approach is to define the Lagrangian, given by

$$L(\boldsymbol{z}, \boldsymbol{\zeta}) = f(\boldsymbol{z}) + \boldsymbol{\zeta}^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{z}), \tag{18}$$

where $\boldsymbol{\zeta} = [\zeta_1, \ldots, \zeta_m]^{\mathrm{T}}$ is the Lagrange multiplier vector. There are two kinds of neurons in LPNN, namely, variable neurons and Lagrangian neurons. The n variable neurons are used to hold the decision variable vector $\boldsymbol{z}$, while the m Lagrangian neurons deal with the Lagrange multiplier vector $\boldsymbol{\zeta}$. In the LPNN framework, the dynamics of the neurons are defined as

$$\frac{d\boldsymbol{z}}{dt} = -\frac{\partial L(\boldsymbol{z}, \boldsymbol{\zeta})}{\partial \boldsymbol{z}}, \tag{19a}$$

$$\frac{d\boldsymbol{\zeta}}{dt} = \frac{\partial L(\boldsymbol{z}, \boldsymbol{\zeta})}{\partial \boldsymbol{\zeta}}. \tag{19b}$$

The differential equations in (19) govern the state transition of the neurons. After the neurons settle down at an equilibrium, the solution is obtained by measuring the neuron outputs at this stable equilibrium point. The purpose of (19 a) is to seek for a state with the minimum objective value, while (19 b) aims at constraining the system state such that it falls into the feasible region. From (19), the network will settle down at a stable state if several mild conditions are satisfied [33,36,37]. It is clear that f and $\boldsymbol{h}$ should be differentiable, otherwise the dynamics cannot be defined.

2.4. Locally competitive algorithm

The LCA, introduced by [41], is also an analog neural network. It is used for handling the following unconstrained optimization problem, given by

$$\min L_{\mathrm{lca}} = \frac{1}{2}\|\boldsymbol{b} - \boldsymbol{\Phi}\boldsymbol{z}\|_2^2 + \lambda\|\boldsymbol{z}\|_1, \tag{20}$$

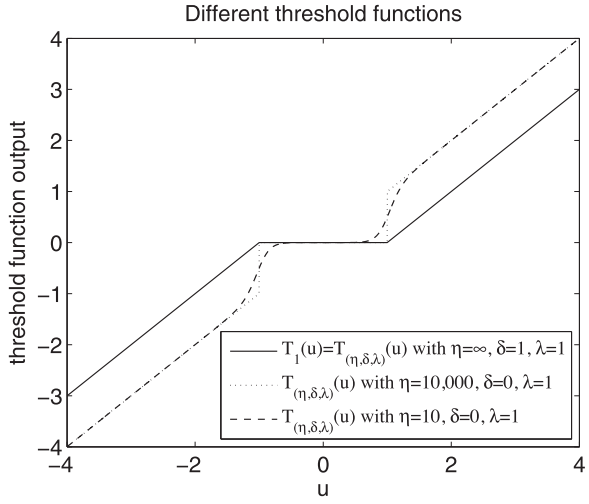


Fig. 1. Examples of general threshold function.

where $\boldsymbol{z} \in \mathbb{R}^n$, $\boldsymbol{b} \in \mathbb{R}^m$ and $\boldsymbol{\Phi} \in \mathbb{R}^{m\times n}$ ($m < n$). For this optimization problem, the LCA uses n neurons to hold the variable vector $\boldsymbol{z}$.

To minimize the cost function L_{lca}, the gradient of L_{lca} should be calculated. Note that the term $\lambda\|\boldsymbol{z}\|_1$ is non-differentiable at zero. In mathematics, the sub-differential, denoted as $\partial\|\boldsymbol{z}\|_1$, can be used to describe the gradient of $\|\boldsymbol{z}\|_1$. Since the sub-differential at a non-differentiable point is equal to a set[1], the implementation of the dynamics becomes infeasible.

The LCA introduces an internal state vector $\boldsymbol{u} = [u_1, \ldots, u_n]^{\mathrm{T}}$ for the neuron output vector $\boldsymbol{z}$. The mapping between $\boldsymbol{z}$ and $\boldsymbol{u}$ is given by

$$z_i = T_\lambda(u_i) = \begin{cases} 0, & |u_i| \le \lambda, \\ u_i - \lambda\,\mathrm{sign}(u_i), & |u_i| > \lambda. \end{cases} \tag{21}$$

In the LCA, $\boldsymbol{z}$ and $\boldsymbol{u}$ are the output state variable and internal state variable vectors, respectively. The parameter λ is a scalar which denotes the threshold of the function.

Furthermore, according to the proof in the appendices of [41], we have

$$\lambda\partial\|\boldsymbol{z}\|_1 \ni \boldsymbol{u} - \boldsymbol{z}. \tag{22}$$

At a non-differentiable point, $\boldsymbol{u} - \boldsymbol{z}$ can be seen as a gradient selection process. The LCA defines its dynamics with respect to $\boldsymbol{u}$ rather than to $\boldsymbol{z}$. Hence, the sub-differentiable term can be replaced according to the relationship given in (22) and we have

$$\frac{d\boldsymbol{u}}{dt} = -\partial_{\boldsymbol{z}} L_{\mathrm{lca}} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{b} - (\boldsymbol{\Phi}^{T}\boldsymbol{\Phi} - \boldsymbol{I})\boldsymbol{z} - \boldsymbol{u}. \tag{23}$$

It should be noticed that if the dynamics of $\boldsymbol{z}$ is used, we need to implement $\partial\|\boldsymbol{z}\|_1$ which is equal to a set for $\forall z_i = 0,\ i = 1, \ldots, n$. In the LCA, for $d\boldsymbol{u}/dt$, the term $\partial\|\boldsymbol{z}\|_1$ can be replaced by $\boldsymbol{u} - \boldsymbol{z}$.

In [41], a more general threshold function was proposed, given by

$$z_i = T_{(\eta,\delta,\lambda)}(u_i) = \mathrm{sign}(u_i)\frac{|u_i| - \delta\lambda}{1 + e^{-\eta(|u_i|-\lambda)}}, \tag{24}$$

where λ still denotes the threshold, η is a parameter to control the speed of the threshold transition and $\delta \in [0, 1]$ indicates what fraction of an additive adjustment is made for values above threshold. Some examples of this general threshold function are provided in Fig. 1. With this threshold function, a more general objective

[1] For the absolute function $|z|$, the sub-differential $\partial|z|$ at $z = 0$ is equal to $[-1, 1]$.

function can be solved, given by

$$\tilde{L}_{\text{lca}} = \frac{1}{2}\|\boldsymbol{b} - \boldsymbol{\Phi z}\|_2^2 + \lambda \sum_{i=1}^{n} \psi_{(\eta,\delta,\lambda)}(z_i). \tag{25}$$

Furthermore, for any $z_i = T_{(\eta,\delta,\lambda)}(u_i)$, the relationship between u_i, z_i and $\partial \psi_{(\eta,\delta,\lambda)}(z_i)/\partial z_i$ is

$$\lambda \frac{\partial \psi_{(\eta,\delta,\lambda)}(z_i)}{\partial z_i} \equiv u_i - z_i. \tag{26}$$

It should be noticed that the analytical expression of $\psi_{(\eta,\delta,\lambda)}(\cdot)$ cannot be obtained generally. However, this does not limit the application of the LCA because the neural dynamics are expressed in terms of the threshold function $T_{(\eta,\delta,\lambda)}(u_i)$ rather than the exact penalty term.

Setting $\eta \to \infty$, $\delta = 0$ and $\lambda = 1$, we obtain an ideal hard threshold function [41], given by

$$z_i = T_{(\infty,0,1)}(u_i) = \begin{cases} 0, & |u_i| \le 1, \\ u_i, & |u_i| > 1. \end{cases} \tag{27}$$

The corresponding penalty term is close to an l_0-norm term, given by

$$\lambda \sum_{i=1}^{n} \psi_{(\infty,0,1)}(z_i) = \frac{1}{2} \sum_{i=1}^{n} \mathcal{I}(|z_i| > 1), \tag{28}$$

where $\mathcal{I}(\cdot)$ is an indicator function. Note that according to (27), the variables z_i produced by the ideal threshold function cannot take values in the range of $[-1, 0)$ and (0,1]. The details of (27) and (28) are provided in [41].

If we set $\eta \to \infty$ and $\delta = 1$, then the general threshold function is reduced to the soft threshold function [41], given by

$$z_i = T_{(\infty,1,\lambda)}(u_i) = T_\lambda(u_i). \tag{29}$$

The corresponding penalty term becomes an l_1-norm function, given by

$$\lambda \sum_{i=1}^{n} \psi_{(\infty,1,\lambda)}(z_i) = \lambda \|\boldsymbol{z}\|_1. \tag{30}$$

The behavior of the dynamics under various settings has been studied in [41,42,47]. However, the limitation of LCA is that it can handle the unconstrained optimization problem only.

3. Development of the proposed algorithm

3.1. Problem formulation

In the CLS method, the l_2-norm is used as its objective function, i.e., $\|\boldsymbol{X}^{\mathrm{T}}\boldsymbol{\alpha}\|_2^2$. It is well known that the l_2-norm works well in Gaussian noise environments, but is sensitive to outliers. In the presence of impulsive noise or outliers, the performance of using the l_p-norm ($p < 2$) is much better than that of using the l_2-norm. Especially, for $p \to 0$, the performance becomes better.

In this study, we formulate the problem as a constrained l_0-norm problem, given by

$$\min_{\boldsymbol{\alpha}} \ \left\|\boldsymbol{X}^{\mathrm{T}}\boldsymbol{\alpha}\right\|_0 \tag{31a}$$

$$\text{s.t. } \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\alpha} = 1, \tag{31b}$$

$$B^2 - 4AC < 0. \tag{31c}$$

We use the LPNN framework to solve the optimization problem stated in (31). Prior to applying the LPNN framework, we need to resolve two issues. First, the inequality constraint in (31) should be convert to an equality, because the LPNN framework can only handle problems with equality constraints. Another issue is that the objective function in (31) is non-differentiable, while the LPNN framework can only solve the problem with differentiable objective and constraints.

To deal with the first issue, we introduce a new variable G and then we can change the inequality constraint, stated in (31 c), to an equality constraint, given by

$$B^2 - 4AC + G^2 = \epsilon, \tag{32}$$

where ϵ is a small negative scalar ($\epsilon = -10^{-12}$ in our experiments).

The formulation of (31) is then modified as

$$\min_{\tilde{\boldsymbol{\alpha}}} \ \left\|\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{\alpha}}\right\|_0 \tag{33a}$$

$$\text{s.t. } \tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\boldsymbol{\Phi}\tilde{\boldsymbol{\alpha}} = 1, \tag{33b}$$

$$\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\boldsymbol{\Theta}\tilde{\boldsymbol{\alpha}} = \epsilon, \tag{33c}$$

where

$$\tilde{\boldsymbol{\alpha}} = [A, B, C, D, E, F, G]^{\mathrm{T}},$$
$$\tilde{\boldsymbol{X}} = [\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2, \ldots, \tilde{\boldsymbol{x}}_N],$$
$$\tilde{\boldsymbol{x}}_i = [x_i^2, x_i y_i, y_i^2, x_i, y_i, 1, 0]^{\mathrm{T}},$$
$$\boldsymbol{\Phi} = \begin{bmatrix} \mathbf{I}_{6\times 6} & \mathbf{0}_{6\times 1} \\ \mathbf{0}_{1\times 6} & 0 \end{bmatrix},$$
$$\boldsymbol{\Theta} = \begin{bmatrix} \boldsymbol{\Lambda} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 1} \\ \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 1} \\ \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} & 1 \end{bmatrix},$$
$$\boldsymbol{\Lambda} = \begin{bmatrix} 0 & 0 & -2 \\ 0 & 1 & 0 \\ -2 & 0 & 0 \end{bmatrix}.$$

The second issue is resolved by considering a general form of the l_p norm. From (24) and (26), we consider the following optimization objective:

$$\min_{\tilde{\boldsymbol{\alpha}},\boldsymbol{z}} \sum_{i=1}^{N} \psi_{(\eta,\delta,\lambda)}([\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{\alpha}}]_i), \tag{34}$$

where $[\cdot]_i$ denotes the ith element of the vector.

The problem stated in (33) becomes

$$\min_{\tilde{\boldsymbol{\alpha}}} \ \sum_{i=1}^{N} \psi_{(\eta,\delta,\lambda)}([\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{\alpha}}]_i) \tag{35a}$$

$$\text{s.t. } \tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\boldsymbol{\Phi}\tilde{\boldsymbol{\alpha}} = 1, \tag{35b}$$

$$\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\boldsymbol{\Theta}\tilde{\boldsymbol{\alpha}} = \epsilon. \tag{35c}$$

If we set $\eta \to \infty$, $\delta = 0$ and $\lambda = 1$ in (35 a), the l_0-norm goal can be achieved.

To exploit the LCA concept, we introduce a dummy vector $\boldsymbol{z}$ and a constraint $\boldsymbol{z} = \tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{\alpha}}$. Therefore, (35) becomes

$$\min_{\tilde{\boldsymbol{\alpha}},\boldsymbol{z}} \sum_{i=1}^{N} \psi_{(\eta,\delta,\lambda)}(z_i) \tag{36a}$$

$$\text{s.t. } \boldsymbol{z} = \tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{\alpha}}, \tag{36b}$$

$$\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\boldsymbol{\Phi}\tilde{\boldsymbol{\alpha}} = 1, \tag{36c}$$

$$\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\boldsymbol{\Theta}\tilde{\boldsymbol{\alpha}} = \epsilon. \tag{36d}$$

3.2. LPNN for ellipse fitting

From (36), we can construct the following Lagrangian function, given by

$$\begin{aligned} &L(\tilde{\boldsymbol{\alpha}}, \boldsymbol{z}, \boldsymbol{\zeta}, \beta, \gamma) \\ &= \sum_{i=1}^{N} \psi_{(\eta,\delta,\lambda)}(z_i) + \boldsymbol{\zeta}^{\mathrm{T}}(\boldsymbol{z} - \tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{\alpha}}) \\ &\quad + \beta(\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\boldsymbol{\Phi}\tilde{\boldsymbol{\alpha}} - 1) + \gamma(\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\boldsymbol{\Theta}\tilde{\boldsymbol{\alpha}} - \epsilon). \end{aligned} \tag{37}$$

In (37), $\tilde{\boldsymbol{\alpha}} \in \mathbb{R}^N$ and $\boldsymbol{z} \in \mathbb{R}^N$ are decision variable vectors, while $\boldsymbol{\zeta} \in \mathbb{R}^N$, β and γ are the Lagrange multipliers. In the next step, we can use (37) to deduce the neural dynamics for the robust ellipse fitting problem given by (36). However, our preliminary experimental results find that the neural dynamics, based on (37), may be unstable.

To improve the stability and convexity, several augmented terms are introduced into the objective function [33–37], then (36) becomes

$$\begin{aligned} &\min_{\tilde{\boldsymbol{\alpha}}, \boldsymbol{z}} \sum_{i=1}^{N} \psi_{(\eta,\delta,\lambda)}(z_i) + \frac{C_0}{2}\left\|\boldsymbol{z} - \tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{\alpha}}\right\|_2^2 \\ &\quad + \frac{C_1}{2}\left(\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\boldsymbol{\Phi}\tilde{\boldsymbol{\alpha}} - 1\right)^2 + \frac{C_2}{2}\left(\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\boldsymbol{\Theta}\tilde{\boldsymbol{\alpha}} - \epsilon\right)^2 \end{aligned} \tag{38a}$$

$$\text{s.t. } \boldsymbol{z} = \tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{\alpha}}, \tag{38b}$$

$$\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\boldsymbol{\Phi}\tilde{\boldsymbol{\alpha}} = 1, \tag{38c}$$

$$\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\boldsymbol{\Theta}\tilde{\boldsymbol{\alpha}} = \epsilon. \tag{38d}$$

In (38), C_0, C_1 and C_2 are positive constants. When they are large enough, the augmented terms [33–37] will make the objective function of (38) to be convex. These three extra terms do not influence the objective function value at an equilibrium point. It is because at an equilibrium point, the constraints are satisfied, i.e., $\boldsymbol{z} = \tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{\alpha}}$, $\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\boldsymbol{\Phi}\tilde{\boldsymbol{\alpha}} = 1$, and $\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\boldsymbol{\Theta}\tilde{\boldsymbol{\alpha}} = \epsilon$. In other words, the values of the augmented terms are equal to zero at an equilibrium point.

With the augmented terms, the Lagrangian for (38) is given by

$$\begin{aligned} &L(\tilde{\boldsymbol{\alpha}}, \boldsymbol{z}, \boldsymbol{\zeta}, \beta, \gamma) \\ &= \sum_{i=1}^{N} \psi_{(\eta,\delta,\lambda)}(z_i) + \boldsymbol{\zeta}^{\mathrm{T}}\left(\boldsymbol{z} - \tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{\alpha}}\right) \\ &\quad + \beta\left(\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\boldsymbol{\Phi}\tilde{\boldsymbol{\alpha}} - 1\right) + \gamma\left(\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\boldsymbol{\Theta}\tilde{\boldsymbol{\alpha}} - \epsilon\right) \\ &\quad + \frac{C_0}{2}\left\|\boldsymbol{z} - \tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{\alpha}}\right\|_2^2 + \frac{C_1}{2}\left(\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\boldsymbol{\Phi}\tilde{\boldsymbol{\alpha}} - 1\right)^2 + \frac{C_2}{2}\left(\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\boldsymbol{\Theta}\tilde{\boldsymbol{\alpha}} - \epsilon\right)^2. \end{aligned} \tag{39}$$

For constructing the neural dynamics, we need to calculate the gradient of Lagrangian (39) with respect to its decision variables and Lagrange variables. To handle the non-differentiable term, we utilize the concept of LCA introducing an internal state variable $\boldsymbol{u}$ for $\boldsymbol{z}$. The relationship between $\boldsymbol{u}$ and $\boldsymbol{z}$ is given by (24), i.e.,

$$z_i = T_{(\eta,\delta,\lambda)}(u_i) = \operatorname{sign}(u_i)\frac{|u_i| - \delta\lambda}{1 + e^{-\eta(|u_i| - \lambda)}}. \tag{40}$$

Now, we define the dynamics on $\boldsymbol{u}$, rather on $\boldsymbol{z}$, given by

$$\frac{du_i}{dt} = -\frac{\partial L(\tilde{\boldsymbol{\alpha}}, \boldsymbol{z}, \boldsymbol{\zeta}, \beta, \gamma)}{\partial z_i}. \tag{41}$$

From (19 a), the dynamics of $\tilde{\boldsymbol{\alpha}}$ are given by

$$\frac{d\tilde{\boldsymbol{\alpha}}}{dt} = -\frac{\partial L(\tilde{\boldsymbol{\alpha}}, \boldsymbol{z}, \boldsymbol{\zeta}, \beta, \gamma)}{\partial \tilde{\boldsymbol{\alpha}}}. \tag{42}$$

From (19 b), for the Lagrangian variables, their dynamics are given by

$$\frac{d\boldsymbol{\zeta}}{dt} = \frac{\partial L(\tilde{\boldsymbol{\alpha}}, \boldsymbol{z}, \boldsymbol{\zeta}, \beta, \gamma)}{\partial \boldsymbol{\zeta}}, \tag{43}$$

$$\frac{d\beta}{dt} = \frac{\partial L(\tilde{\boldsymbol{\alpha}}, \boldsymbol{z}, \boldsymbol{\zeta}, \beta, \gamma)}{\partial \beta}, \tag{44}$$

$$\frac{d\gamma}{dt} = \frac{\partial L(\tilde{\boldsymbol{\alpha}}, \boldsymbol{z}, \boldsymbol{\zeta}, \beta, \gamma)}{\partial \gamma}. \tag{45}$$

According to (26) and (39), the dynamics given by (41)–(45) become

$$\frac{d\boldsymbol{u}}{dt} = -\boldsymbol{u} + \boldsymbol{z} - \boldsymbol{\zeta} - C_0\left(\boldsymbol{z} - \tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{\alpha}}\right), \tag{46}$$

$$\begin{aligned} \frac{d\tilde{\boldsymbol{\alpha}}}{dt} &= \tilde{\boldsymbol{X}}\boldsymbol{\zeta} - 2\beta\boldsymbol{\Phi}\tilde{\boldsymbol{\alpha}} - 2\gamma\boldsymbol{\Theta}\tilde{\boldsymbol{\alpha}} - C_0\tilde{\boldsymbol{X}}\left(\boldsymbol{z} - \tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{\alpha}}\right) \\ &\quad - 2C_1\left(\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\boldsymbol{\Phi}\tilde{\boldsymbol{\alpha}} - 1\right)\boldsymbol{\Phi}\tilde{\boldsymbol{\alpha}} - 2C_2\left(\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\boldsymbol{\Theta}\tilde{\boldsymbol{\alpha}} - \epsilon\right)\boldsymbol{\Theta}\tilde{\boldsymbol{\alpha}}, \end{aligned} \tag{47}$$

$$\frac{d\boldsymbol{\zeta}}{dt} = \boldsymbol{z} - \tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{\alpha}}, \tag{48}$$

$$\frac{d\beta}{dt} = \tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\boldsymbol{\Phi}\tilde{\boldsymbol{\alpha}} - 1, \tag{49}$$

$$\frac{d\gamma}{dt} = \tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\boldsymbol{\Theta}\tilde{\boldsymbol{\alpha}} - \epsilon. \tag{50}$$

It should be noticed that for our formulation, we should set η as a large number, $\delta = 0$ and $\lambda = 1$ in (24).

3.3. Properties and simulation method

In the LPNN approach, the circuit complexity depends on the time derivative calculations. From (46)–(50), the most computationally demanding step is to determine the product of an $N \times 7$ matrix and 7×1 vector. Hence the complexity to obtain the time derivatives is equal to $\mathcal{O}(N)$ only.

Upon convergence of the iterative procedure, we obtain the estimate of $\tilde{\boldsymbol{\alpha}}$, denoted by $\tilde{\boldsymbol{\alpha}}^*$. From $\tilde{\boldsymbol{\alpha}}^*$, the ellipse parameter estimates $\{a^*, b^*, c_x^*, c_y^*, \theta^*\}$ are then computed from:

$$\theta^* = \frac{1}{2}\tan^{-1}\left(\frac{\tilde{\alpha}_2^*}{\tilde{\alpha}_1^* - \tilde{\alpha}_3^*}\right), \tag{51}$$

$$\begin{bmatrix} c_x^* \\ c_y^* \end{bmatrix} = \begin{bmatrix} -2\tilde{\alpha}_1^* - \tilde{\alpha}_2^* \\ -\tilde{\alpha}_2^* - 2\tilde{\alpha}_3^* \end{bmatrix}^{-1} \begin{bmatrix} \tilde{\alpha}_4^* \\ \tilde{\alpha}_5^* \end{bmatrix}, \tag{52}$$

$$a^* = \sqrt{\frac{\begin{bmatrix} c_x^* \\ c_y^* \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \tilde{\alpha}_1^* & \tilde{\alpha}_2^*/2 \\ \tilde{\alpha}_2^*/2 & \tilde{\alpha}_3^* \end{bmatrix} \begin{bmatrix} c_x^* \\ c_y^* \end{bmatrix} + 1}{\mu_1}}, \tag{53}$$

$$b^* = \sqrt{\frac{\begin{bmatrix} c_x^* \\ c_y^* \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \tilde{\alpha}_1^* & \tilde{\alpha}_2^*/2 \\ \tilde{\alpha}_2^*/2 & \tilde{\alpha}_3^* \end{bmatrix} \begin{bmatrix} c_x^* \\ c_y^* \end{bmatrix} + 1}{\mu_2}}, \tag{54}$$

where $\mu_1 = \tilde{\alpha}_1^*\cos^2\theta^* + \tilde{\alpha}_2^*\sin\theta^*\cos\theta^* + \tilde{\alpha}_3^*\sin^2\theta^*$ and $\mu_2 = \tilde{\alpha}_1^*\sin^2\theta^* - \tilde{\alpha}_2^*\sin\theta^*\cos\theta^* + \tilde{\alpha}_3^*\cos^2\theta^*$. Fig. 2 shows the dynamics of the estimated parameters in a typical experiment. The settings are described in Section 5.2. It is seen that the network can settle down within 40 characteristic times.

In the simulation section, we use a discrete method to simulate the dynamics. The dynamics, (46)–(50), are discretized as

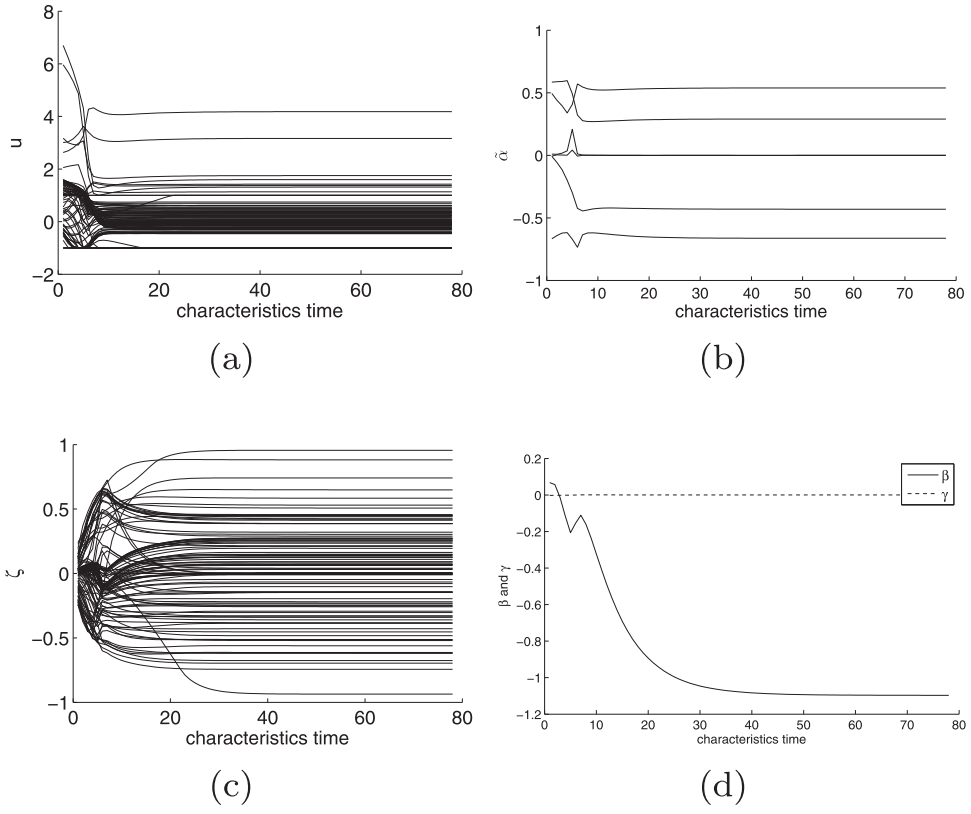


Fig. 2. Dynamics of estimated parameters under Laplacian noise when the noise level is $0.8\sqrt{2}$. (a) $\mathbf{u}$; (b) $\tilde{\boldsymbol{\alpha}}$; (c) $\boldsymbol{\zeta}$; (d) β and γ.

$$\boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(k)} + \mu \frac{d\boldsymbol{u}^{(k)}}{dt}, \tag{55}$$

$$\tilde{\boldsymbol{\alpha}}^{(k+1)} = \tilde{\boldsymbol{\alpha}}^{(k)} + \mu \frac{d\tilde{\boldsymbol{\alpha}}^{(k)}}{dt}, \tag{56}$$

$$\boldsymbol{\zeta}^{(k+1)} = \boldsymbol{\zeta}^{(k)} + \mu \frac{d\boldsymbol{\zeta}^{(k)}}{dt}, \tag{57}$$

$$\beta^{(k+1)} = \beta^{(k)} + \mu \frac{d\beta^{(k)}}{dt}, \tag{58}$$

$$\gamma^{(k+1)} = \gamma^{(k)} + \mu \frac{d\gamma^{(k)}}{dt}, \tag{59}$$

where the superscript (k) denotes the kth iteration and $\mu > 0$ is a small positive constant.

4. Stability of the proposed algorithm

For an analog neural network, the stability of its dynamics is a crucial property that needs to be investigated. For the ellipse fitting model shown in (38), its global stability is hard to be proved. This section discusses the local stability of the proposed model. That means, an equilibrium point should be stable. Otherwise, the network can never converge to it.

Let $\{\tilde{\boldsymbol{\alpha}}^*, \boldsymbol{u}^*, \boldsymbol{\zeta}^*, \beta^*, \gamma^*\}$ be an equilibrium point of the dynamics given by (46)–(50). Let $\tilde{\boldsymbol{\alpha}}^*, \boldsymbol{u}^*$ be the corresponding state variable vectors. There are two sufficient conditions for local stability in the LPNN approach. The first one is that the Hessian matrix of the Lagrangian (39) at $\{\tilde{\boldsymbol{\alpha}}^*, \boldsymbol{u}^*, \boldsymbol{\zeta}^*, \beta^*, \gamma^*\}$ should be positive definite. It has been achieved by introducing the augmented terms. Because according to [33–37], as long as the augmented terms are large enough, at an equilibrium point, the Hessian is positive definite under mild conditions.

The second condition is that at an equilibrium point, the gradient vectors of the constraints with respect to the state variables should be linearly independent. In (38), we have $N+2$ constraints given by

$$h_1(\tilde{\boldsymbol{\alpha}}, \boldsymbol{z}) = \tilde{\boldsymbol{\alpha}}^{\mathrm{T}} \boldsymbol{\Phi} \tilde{\boldsymbol{\alpha}} - 1, \tag{60}$$

$$h_2(\tilde{\boldsymbol{\alpha}}, \boldsymbol{z}) = \tilde{\boldsymbol{\alpha}}^{\mathrm{T}} \boldsymbol{\Theta} \tilde{\boldsymbol{\alpha}} - \epsilon, \tag{61}$$

$$h_{i+2}(\tilde{\boldsymbol{\alpha}}, \boldsymbol{z}) = z_i - \tilde{\boldsymbol{\alpha}}^{\mathrm{T}} \tilde{\boldsymbol{x}}_i, \quad i = 1, \ldots, N. \tag{62}$$

The gradient vectors with respect to $\{\tilde{\boldsymbol{\alpha}}^*, \boldsymbol{u}^*\}$ are given by

$$\left\{ \begin{bmatrix} \frac{\partial h_1(\tilde{\boldsymbol{\alpha}}^*, \boldsymbol{z}^*)}{\partial \tilde{\boldsymbol{\alpha}}} \\ \frac{\partial h_1(\tilde{\boldsymbol{\alpha}}^*, \boldsymbol{z}^*)}{\partial \boldsymbol{u}} \end{bmatrix}, \cdots, \begin{bmatrix} \frac{\partial h_{N+2}(\tilde{\boldsymbol{\alpha}}^*, \boldsymbol{z}^*)}{\partial \tilde{\boldsymbol{\alpha}}} \\ \frac{\partial h_{N+2}(\tilde{\boldsymbol{\alpha}}^*, \boldsymbol{z}^*)}{\partial \boldsymbol{u}} \end{bmatrix} \right\}$$

$$= \left\{ \begin{bmatrix} 2A \\ 2B \\ 2C \\ 2D \\ 2E \\ 2F \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} -2C \\ B \\ -2A \\ 0 \\ 0 \\ 0 \\ G \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} -x_1^2 \\ -x_1 y_1 \\ -y_1^2 \\ -x_1 \\ -y_1 \\ -1 \\ 0 \\ g_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \cdots, \begin{bmatrix} -x_N^2 \\ -x_N y_N \\ -y_N^2 \\ -x_N \\ -y_N \\ -1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ g_N \end{bmatrix} \right\} \tag{63}$$

where

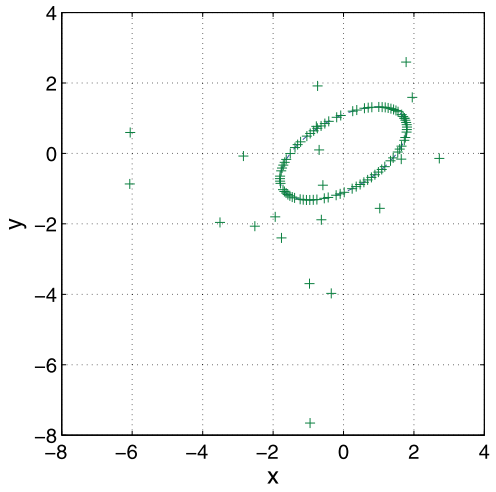
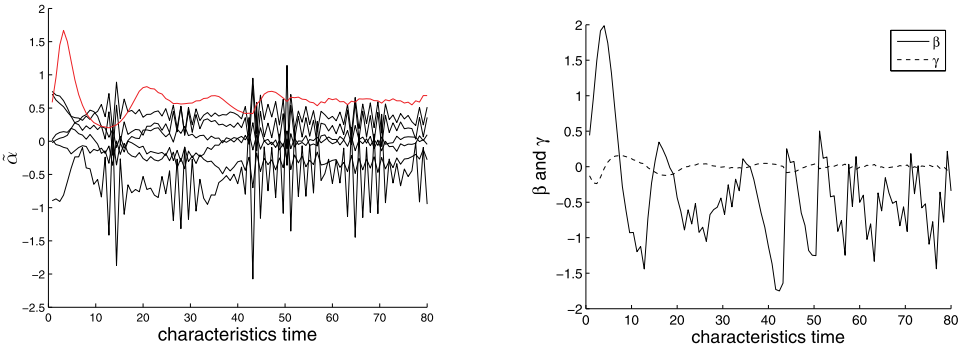


Fig. 3. Ellipse data with 20 scattering points contaminated by Laplacian noise.

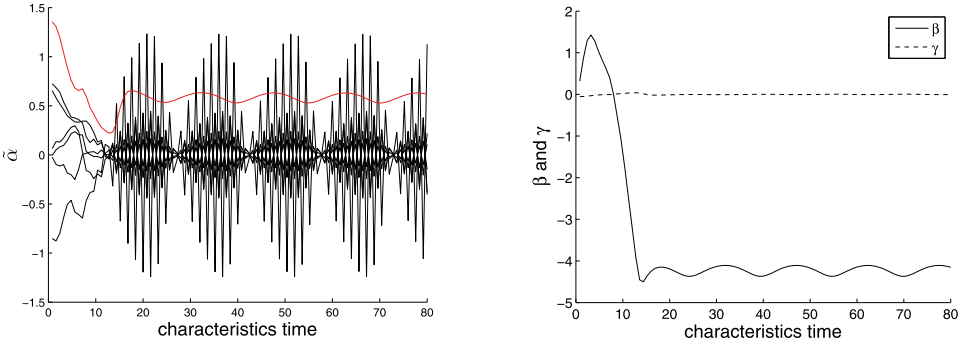
$$g_i = \frac{\partial h_{i+2}(\tilde{\boldsymbol{\alpha}}, \boldsymbol{z})}{\partial z_i}\frac{\partial z_i}{\partial u_i} = \frac{1}{1+\exp\left(-\eta(|u_i|-\lambda)\right)} + \frac{\eta(|u_i|-\delta\lambda)\exp\left(-\eta(|u_i|-\lambda)\right)}{(1+\exp\left(-\eta(|u_i|-\lambda)\right))^2}.$$

For the proposed approach, we set η to be a large positive number, $\delta = 0$, and $\lambda = 1$. Hence it is easy to show that g_i is positive for $\forall i = 1, \ldots, N$.

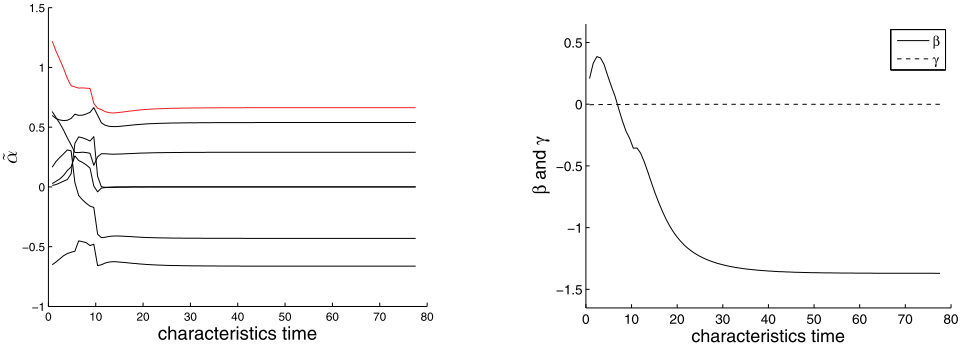
In (63), there are $N+2$ gradient vectors and each of them has $N+7$ elements. It is easy to note that the last N vectors are linearly independent with each other. Besides, they are all linearly independent with the first two vectors. At an equilibrium, if the estimated G is not equal to zero, then all the $N+2$ gradient vectors are linearly independent. Therefore, $\{\tilde{\boldsymbol{\alpha}}^*, \boldsymbol{u}^*, \boldsymbol{\zeta}^*, \beta^*, \gamma^*\}$ is an asymptotically stable point of the neural network, if the estimated G, i.e., $\tilde{\alpha}_7$ is not equal to zero. For any points nearby the equilibrium point $\{\tilde{\boldsymbol{\alpha}}^*, \boldsymbol{u}^*, \boldsymbol{\zeta}^*, \beta^*, \gamma^*\}$, the state of the network must converge to this equilibrium.



(a) $\{C_0, C_1, C_2\} = \{0.05, 0.1, 0.1\}$



(b) $\{C_0, C_1, C_2\} = \{0.5, 1, 1\}$



(c) $\{C_0, C_1, C_2\} = \{5, 10, 10\}$

Fig. 4. Dynamics of the estimated parameters $\tilde{\boldsymbol{\alpha}}$, β and γ with different values of $\{C_0, C_1, C_2\}$. The first column represents the dynamics of $\tilde{\boldsymbol{\alpha}}$, while the second is the dynamics of β and γ. The Laplacian noise level is $0.8\sqrt{2}$.

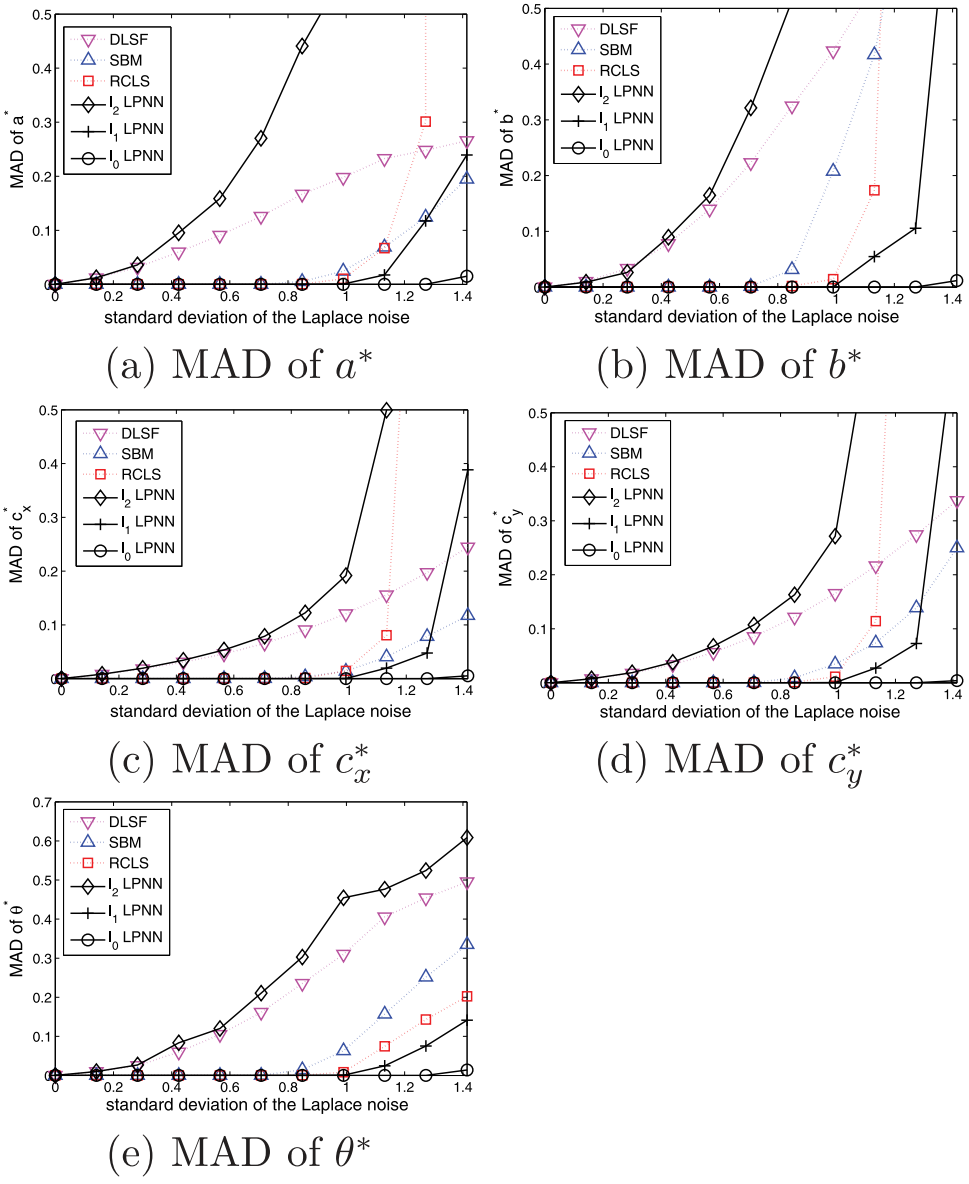


Fig. 5. MAD results of various algorithms in Laplacian noise. The Laplacian noise level is varied from 0 to $\sqrt{2}$.

5. Simulations

This section conducts several experiments to evaluate the performance of the proposed l_0-norm LPNN approach. Several state-of-the-art ellipse fitting algorithms are implemented for performance comparison. They are the direct least squares fitting (DLSF) [46], SBM [16], and RCLS [17]. Note that for the DLSF algorithm, it solves a generalized eigenvalue problem to fit an ellipse. The SBM method [16] introduces two regularized terms and determines ellipse parameters by solving a second-order cone programming (SOCP) problem. The RCLS algorithm combines the maximum correntropy criterion with the CLS method. In addition, as comparison, we also apply LPNN to solve the l_2-norm and the l_1-norm based formulation.

For the LPNN approach, we consider three approaches. One is our proposed l_0-norm approach, namely ***l_0*-LPNN**. For the proposed l_0-LPNN, we set $\eta = 10,000$, $\delta = 0$ and $\lambda = 1$, and the threshold is

$$z_i = T_{(10000,0,1)}(u_i) = \text{sign}(u_i)\frac{|u_i|}{1+e^{-10000(|u_i|-1)}}. \tag{64}$$

Another one is the l_1-norm approach, in which we set $\eta \to \infty$, $\delta = 1$ and $\lambda = 1$. The threshold is given by

$$z_i = T_1(u_i) = \begin{cases} 0, & |u_i| \le 1, \\ u_i - \text{sign}(u_i), & |u_i| > 1. \end{cases} \tag{65}$$

For the l_2-norm version, we apply the LPNN directly to solve:

$$\min_{\boldsymbol{\alpha},\boldsymbol{z}} \quad \|\boldsymbol{z}\|_2^2 \tag{66a}$$

$$\text{s.t. } \boldsymbol{z} = \boldsymbol{X}^{\mathrm{T}}\tilde{\boldsymbol{\alpha}}, \tag{66b}$$

$$\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\boldsymbol{\Phi}\tilde{\boldsymbol{\alpha}} = 1, \tag{66c}$$

$$\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\boldsymbol{\Theta}\tilde{\boldsymbol{\alpha}} = \epsilon. \tag{66d}$$

It is expected that (66) is just an alternative implementation of the CLS estimator in [15] with an additional constraint to make sure the fitting result is an ellipse.

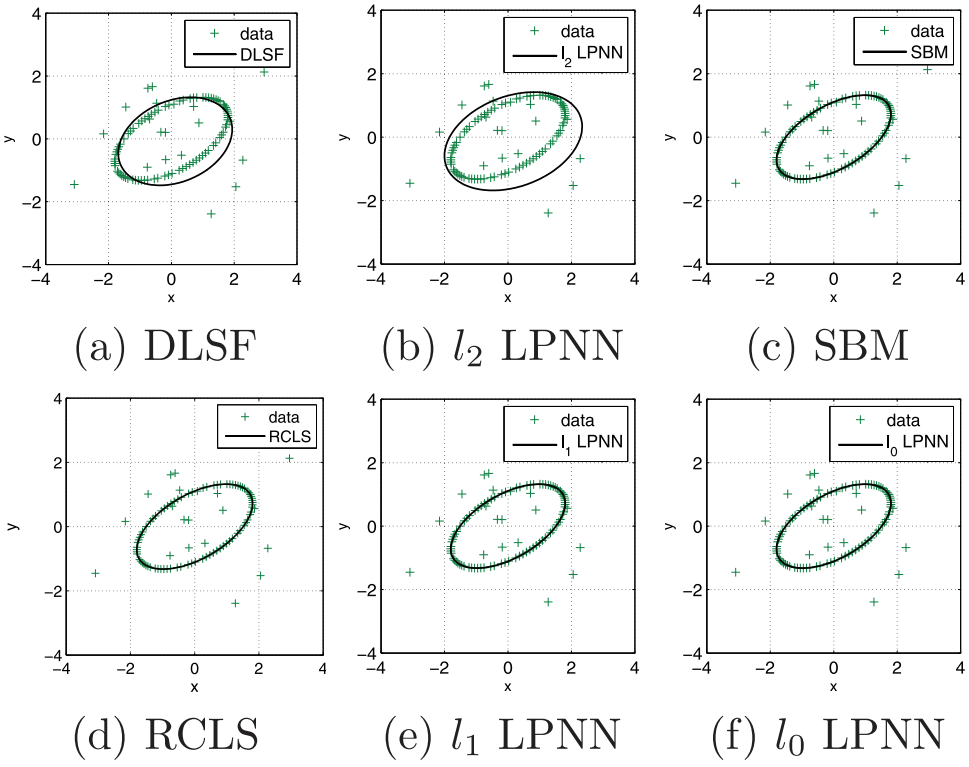


Fig. 6. Fitting results of a typical run at Laplacian noise level of $0.7\sqrt{2}$ (around 0.9899).

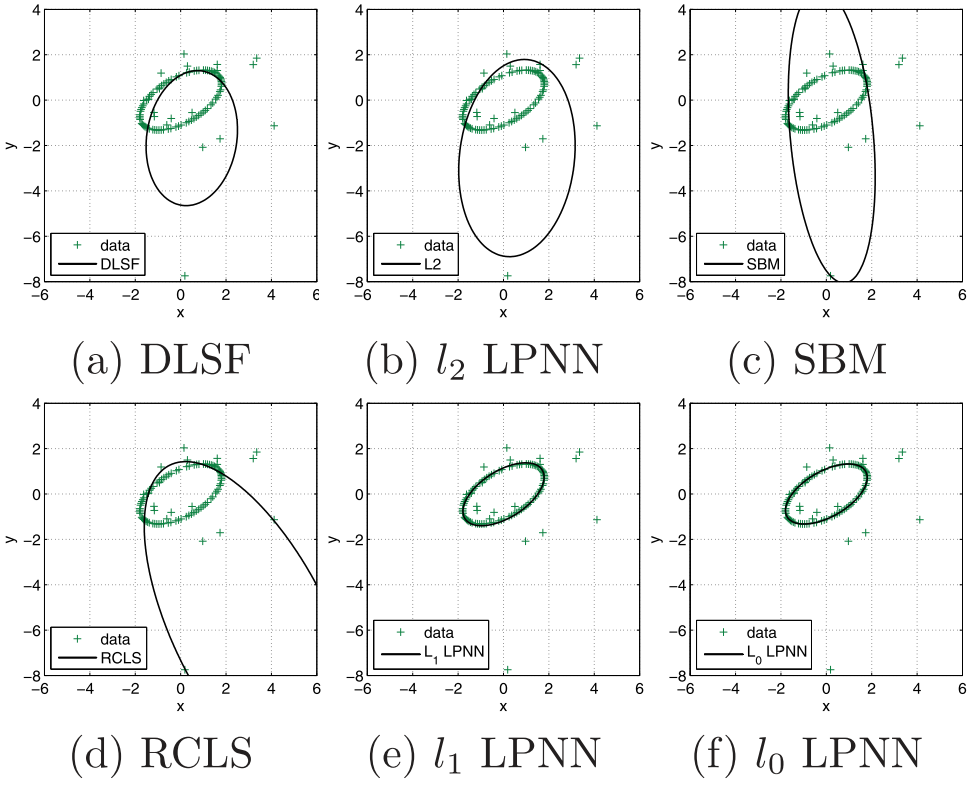


Fig. 7. Fitting results of a typical run at Laplacian noise level of $0.9\sqrt{2}$ (around 1.2728).

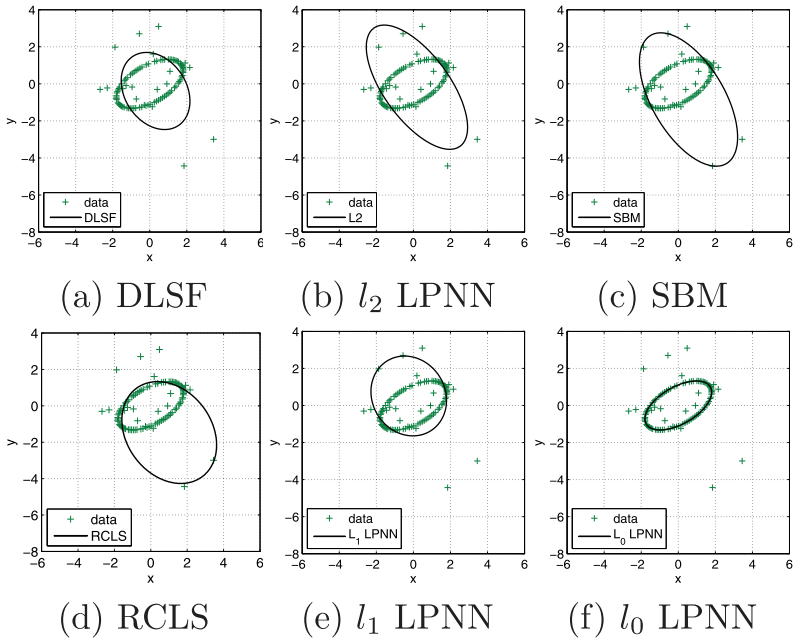


Fig. 8. Fitting results of a typical run at Laplacian noise level of $\sqrt{2}$ (around 1.4142).

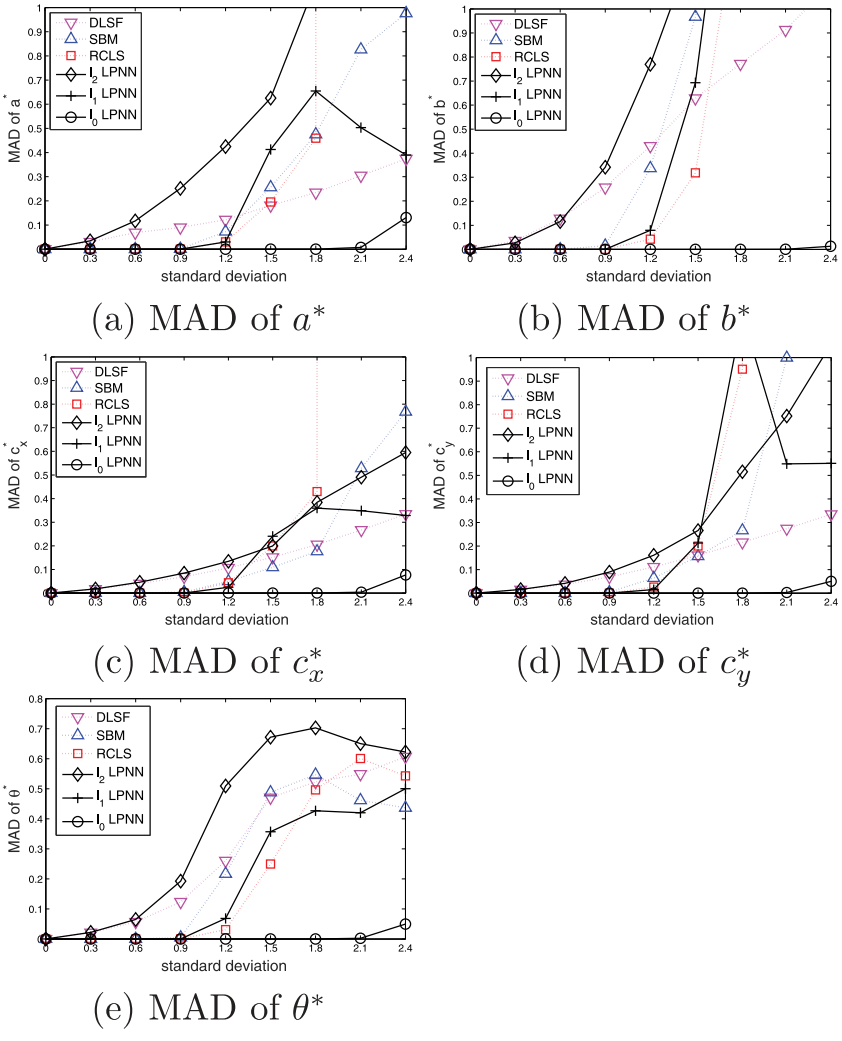


Fig. 9. The MAD results of various algorithms in uniform noise. The uniform noise level is varied from 0 to 2.4.

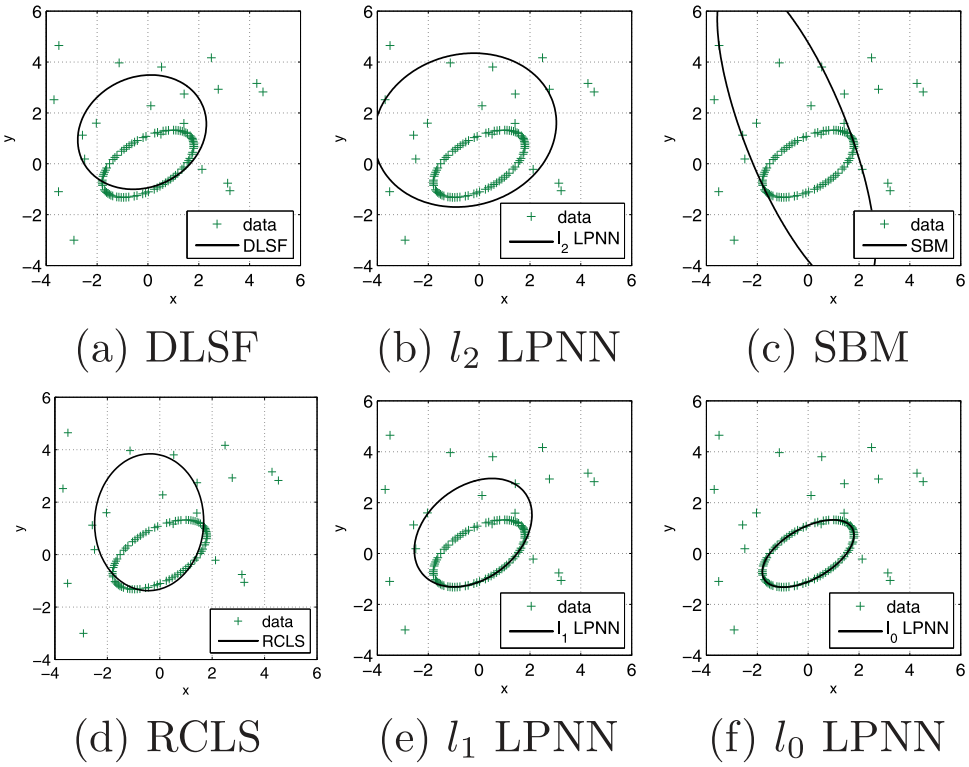


Fig. 10. The fitting results of a typical run for the uniform noise level equal to 2.4.

In the proposed LPNN approach, C_0, C_1, C_2 are three tunable parameters and we use trial-and-error method to select them. We try 6 C_0 values: $C_0 = \{1, 2, 3, 4, 5, 6\}$ and 6 C_1, C_2 values: $C_1 = C_2 = \{2, 4, 6, 8, 10, 12\}$, and finally choose $C_0 = 5, C_1 = 10, C_2 = 10$. For ϵ, we select $\epsilon = -10^{-12}$.

In the discrete simulation, the step size μ is selected as 0.0001. We also need to initialize the state variables $\tilde{\boldsymbol{\alpha}}$ and $\boldsymbol{u}$, and the Lagrangian variables $\boldsymbol{\zeta}$, β and γ. The $\tilde{\boldsymbol{\alpha}}$ is not initialized with the CLS method because its solution may not correspond to an ellipse. Instead, we compute the initial estimate of $\tilde{\boldsymbol{\alpha}}$ by assuming that the data points are sampled from a circle. That is, the circle center is given by the midpoint of the data set while the radius is a small positive random value. Once the circle is constructed, it is easy to initialize $\tilde{\boldsymbol{\alpha}}$. We can also get initial estimates of $\mathbf{u}$ by $\mathbf{u} = \tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{\alpha}}$. The initial values of the Lagrangian variables $\boldsymbol{\zeta}$, β and γ are small random values.

5.1. Stability and convergence

In this subsection, we show the stability of our proposed algorithm. The settings are described in Section 5.2. As mentioned in Section 4, the augmented terms should be large enough. That is, C_0, C_1, and C_2 should be sufficiently large. To illustrate the stability and convergence of our algorithm, we test three settings of $\{C_0, C_1, C_2\}$: {0.05, 0.1, 0.1}, {0.5, 1, 1}, and {5, 10, 10}. The dynamics of the networks under these three settings are shown in Fig. 4. It can be seen that when $\{C_0, C_1, C_2\}$ are with small values, their dynamics fluctuate and the states of the neurons do not converge, as shown in the first two rows of Fig. 4. When $\{C_0, C_1, C_2\} = \{5, 10, 10\}$, the states of the neurons converge within around 80 characteristic times. In addition, the estimated G (red curve) is always bigger than 0 for all the three settings, as shown in Fig. 4. In our examples, only the dynamics with $\{C_0, C_1, C_2\} = \{5, 10, 10\}$ converge.

5.2. Ellipse fitting in Laplacian noise

In this experiment, we test the performance of our proposed approach in different Laplacian noise levels. Firstly, we generate an ellipse with 100 data points, which is shown in Fig. 3. The true elliptical parameters are $c_x = 0$, $c_y = 0$, $a = 2$, $b = 1$, $\theta = 30^\circ$. We add small Gaussian noise with variance 10^{-8} to these points. We then randomly choose 20 points from the data set and add zero-mean Laplacian noise into them, which are also illustrated in Fig. 3. The standard deviation of the Laplacian noise is varied from 0 to $\sqrt{2}$. We repeat the experiment 100 times at each noise level and compute the mean absolute deviation (MAD) of the estimated parameters (c_x^*, c_y^*, a^*, b^*, θ^*). The results are shown in Fig. 5. It can be seen that the l_2-norm LPNN and DLSF algorithms are very sensitive to outliers. The SBM and RCLS methods can effectively decrease the impact of outliers. However, both of them start to break down when the Laplacian noise level is greater than 0.9899. For the l_1-norm LPNN, it works well until the noise level is 1.1314. Furthermore, the l_0-norm LPNN still works very well up to the noise level of $\sqrt{2}$.

Fig. 6 shows the fitting results of a typical run when the noise level is equal to 0.9899. It can be seen that the l_2-norm LPNN and DLSF methods do not offer reliable results, while the remaining algorithms can provide a satisfactory fitting. Fig. 7 plots the fitting results of a typical run at the noise level of 1.2728. We observe that only the l_1-norm and l_0-norm LPNN algorithms can achieve an accurate ellipse fitting. When we increase the noise level to 1.4142, only the l_0-norm LPNN algorithm works well, which is shown in Fig. 8.

5.3. Ellipse fitting in uniform noise

In the second experiment, we test the performance of various algorithms under uniform noise. The experimental setting is the

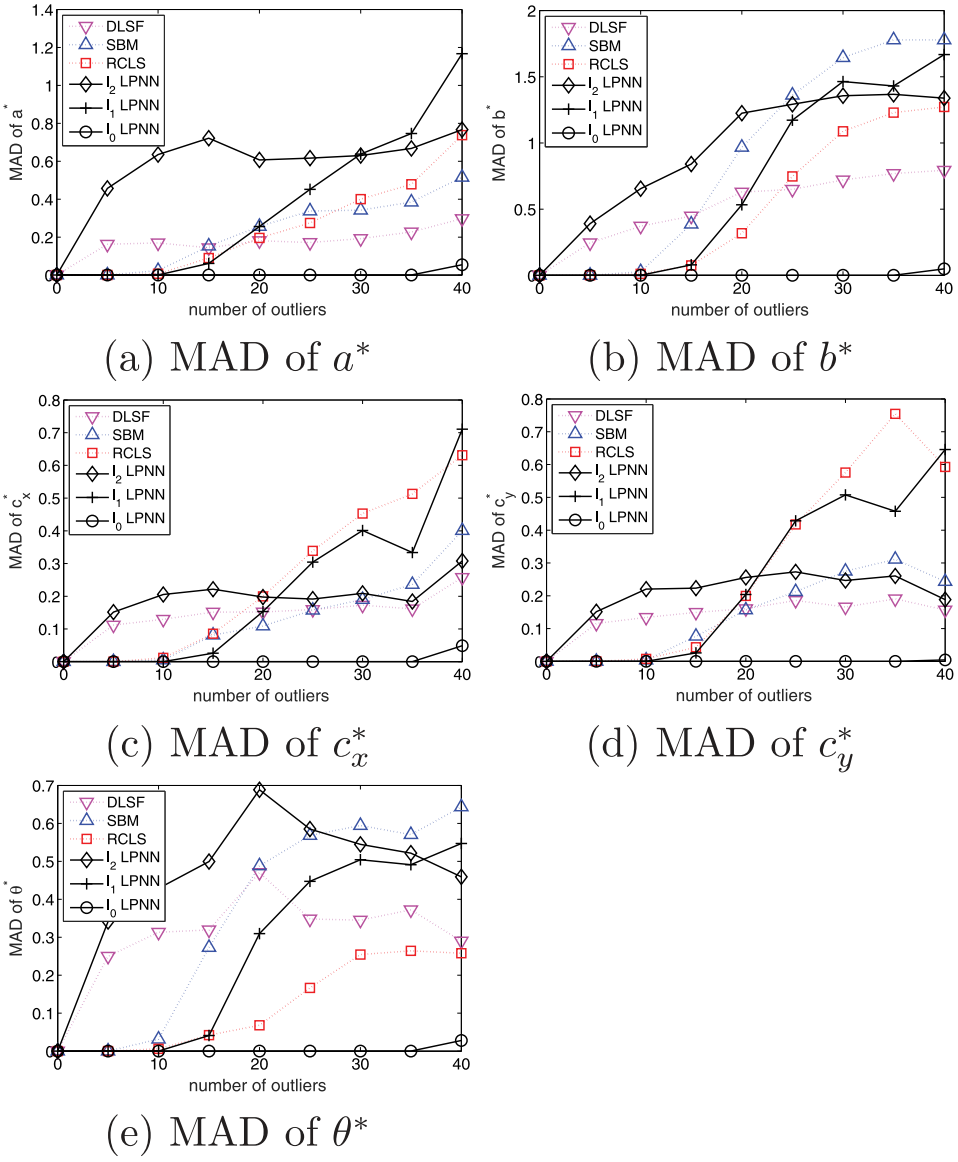


Fig. 11. The MAD results of different algorithms in uniform noise. The uniform noise level is fixed at 1.5, but number of noisy points changes from 0 to 40.

same as Section 5.2, except that the Laplacian noise is replaced by the uniform noise. The noise standard deviation is now varied from 0 to 2.4. To compute the MAD of the estimated parameters, we repeat the experiment 100 times at each noise level. The results are shown in Fig. 9. It is observed that the l_2-norm LPNN and DLSF algorithms are very sensitive to outliers. The SBM, RCLS and l_1-norm LPNN methods start to break down when the uniform noise level is around 0.9 to 1.2. The l_0-norm LPNN still works very well up to the noise level of 2.4. Fig. 10 shows the fitting results of a typical run at the noise level of 2.4. It can be seen that only the l_0-norm LPNN method produces a satisfactory fitting result.

5.4. Ellipse fitting with different number of noisy data points

In the third experiment, we fix the standard deviation of the uniform noise at 1.5, but change the number of noisy points from 0 to 40. Other settings are the same as Section 5.3. We repeat the experiment 100 times at each setting. The results are shown in Fig. 11. The l_2-norm LPNN and DLSF algorithms are very sensitive to the quantity of outliers. The SBM, RCLS and l_1-norm LPNN methods cannot work when the number of noisy points is larger than 10. The l_0-norm LPNN can give satisfactory results until the number of noisy points is 40.

5.5. Real data with pepper noise

In the fourth experiment, we test the performance of various algorithms with real data.

Fig. 12(a) shows a human eye image [17] and this kind of images is frequently used in iris recognition where a key step is to find out the correct pupil region. In this test, our target is to fit the pupil region of the eye. After edge extraction, Fig. 12(b) is obtained. For the extracted image, we randomly add some pepper noise whose density is 0.001. The observations are provided in Fig. 12(c). Finally, we apply various robust ellipse fitting algorithms including SBM, RCLS, l_1-norm LPNN, and l_0-norm LPNN to the data and the fitting results are given by Fig. 12 (d)–12 (g). We can see that the RCLS and SBM both are influenced by the pepper noises, but l_1-norm LPNN and l_0-norm LPNN give out satisfactory results.

Fig. 13(a) shows a real image of space probe [17] and here the task is to fit the circumference of the antenna. After edge detection, Fig. 13 (b) is obtained. Same as the process mentioned

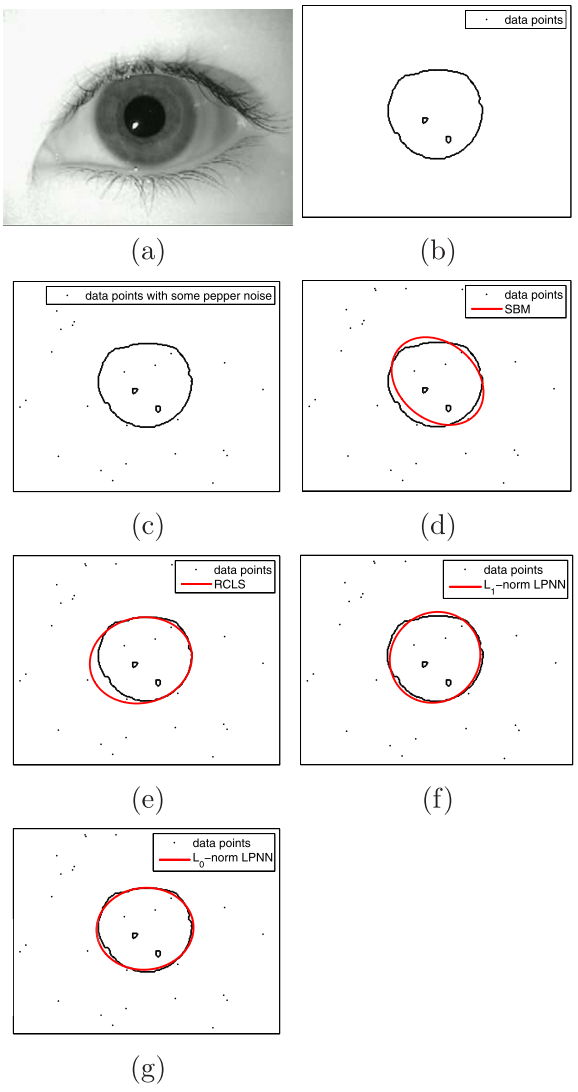


Fig. 12. Fitting results of a human eye image. (a) Actual image. (b) Data points after edge extraction. (c) Observations with pepper noise. (d) Fitting result of SBM. (e) Fitting result of RCLS. (f) Fitting result of l_1-norm LPNN. (g) Fitting result of l_0-norm LPNN.

before, we add pepper noise whose density is 0.001. The resultant observed data are given in Fig. 13(c). Fig. 13 (d) Fig. 13(g) shows the fitting results of the SBM, RCLS, l_1-norm LPNN, and l_0-norm LPNN. It can be seen that the SBM, RCLS and l_1-norm LPNN do not work very well. Although the l_1-norm can suppress the effect of outliers, the fitting result of l_1-norm LPNN method is worse than the result of l_0-norm LPNN scheme.

6. Conclusion

Many applications require fitting 2-D noisy data points with an ellipse. To reduce the influence of outliers, this paper proposes a robust ellipse fitting approach based on the concept of LPNN. Inspired by the properties of l_0-norm, we redesign the objective function of the original ellipse fitting problem to make it robust against impulsive noise and outliers. Since the conventional LPNN is able to handle differentiable objective functions only, we introduce the LCA concept into the LPNN framework. It is demonstrated that the proposed l_0-norm LPNN method can effectively reduce the influence of outliers and is better than other robust ellipse fitting algorithms.

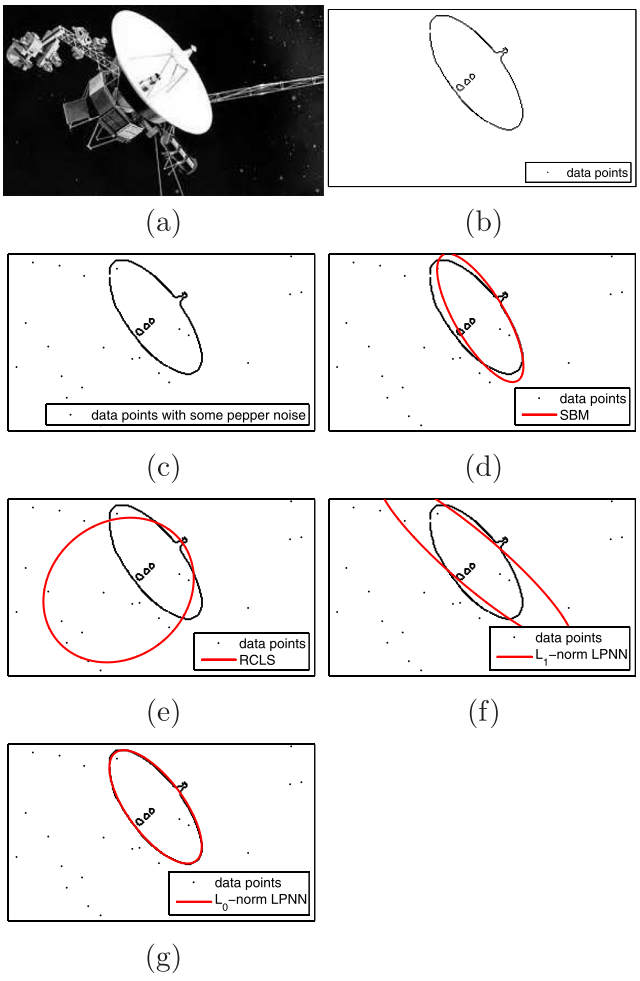


Fig. 13. Fitting results of a space probe image. (a) Actual image. (b) Data points after edge extraction. (c) Observations with pepper noise. (d) Fitting result of SBM. (e) Fitting result of RCLS. (f) Fitting result of l_1-norm LPNN. (g) Fitting result of l_0-norm LPNN.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Zhanglei Shi: Conceptualization, Methodology, Software, Writing - original draft. **Hao Wang:** Conceptualization, Methodology, Software, Writing - original draft. **Chi-Sing Leung:** Supervision, Writing - review & editing. **Hing Cheung So:** Supervision, Writing - review & editing. **Kim-Fung Tsang:** Writing - review & editing. **Anthony G. Constantinides:** Methodology.

Acknowledgment

This work was supported by the General Research Fund from The Government of the Hong Kong Special Administrative Region under Grant CityU 11259516 and the ITF from the Government of the Hong Kong Special Administrative Region under Grant ITP/058/17LP.

References

[1] N. Chernov, G. Ososkov, Effective algorithms for circle fitting, Comput. Phys. Commun. 33 (4) (1984) 329–333.
[2] K. Paton, Conic sections in chromosome analysis, Pattern Recognit. 2 (1) (1970) 39–51.
[3] M. Liao, Y. qian Zhao, X. hua Li, P. shan Dai, X. wen Xu, J. kai Zhang, B. ji Zou, Automatic segmentation for cell images based on bottleneck detection and ellipse fitting, Neurocomputing 173 (2016) 615–622.
[4] X. Ben, W. Meng, R. Yan, Dual-ellipse fitting approach for robust gait periodicity detection, Neurocomputing 79 (2012) 173–178.
[5] J. Qi, P. Yang, M. Hanneghan, S. Tang, Multiple density maps information fusion for effectively assessing intensity pattern of lifelogging physical activity, Neurocomputing 220 (2017) 199–209.
[6] C.-F. Juang, S.-J. Shiu, Using self-organizing fuzzy network with support vector learning for face detection in color images, Neurocomputing 71 (16) (2008) 3409–3420.
[7] R.O. Duda, P.E. Hart, Use of the Hough transformation to detect lines and curves in pictures, Commun. ACM 15 (1) (1972) 11–15.
[8] D.H. Ballard, Generalizing the Hough transform to detect arbitrary shapes, Pattern Recognit. 13 (2) (1981) 111–122.
[9] D. Barwick, Very fast best-fit circular and elliptical boundaries by chord data, IEEE Trans. Pattern Anal. Mach. Intell. 31 (6) (2009) 1147–1152.
[10] Y. Nakagawa, A. Rosenfeld, A note on polygonal and elliptical approximation of mechanical parts, Pattern Recognit. 11 (2) (1979) 133–142.
[11] P.L. Rosin, G.A. West, Nonparametric segmentation of curves into various representations, IEEE Trans. Pattern Anal. Mach. Intell. (12) (1995) 1140–1153.
[12] F.L. Bookstein, Fitting conic sections to scattered data, Comput. Graph. Image Process. 9 (1) (1979) 56–71.
[13] K. Kanatani, Statistical bias of conic fitting and renormalization, IEEE Trans. Pattern Anal. Mach. Intell. (3) (1994) 320–326.
[14] P.D. Sampson, Fitting conic sections to ẗvery scatteredg data: an iterative refinement of the Bookstein algorithm, Comput. Graph. Image Process. 18 (1) (1982) 97–108.
[15] W. Gander, G.H. Golub, R. Strebel, Least-squares fitting of circles and ellipses, BIT Numer. Math. 34 (4) (1994) 558–578.
[16] J. Liang, M. Zhang, D. Liu, X. Zeng, O. Ojowu, K. Zhao, Z. Li, H. Liu, Robust ellipse fitting based on sparse combination of data points, IEEE Trans. Image Process. 22 (6) (2013) 2207–2218.
[17] J. Liang, Y. Wang, X. Zeng, Robust ellipse fitting via half-quadratic and semidefinite relaxation optimization, IEEE Trans. Image Process. 24 (11) (2015) 4276–4286.
[18] D. Tank, J. Hopfield, Simple'neural'optimization networks: an a/d converter, signal decision circuit, and a linear programming circuit, IEEE Trans. Circuits Syst. 33 (5) (1986) 533–541.
[19] L. Chua, G.-N. Lin, Nonlinear programming without computation, IEEE Trans. Circuits Syst. 31 (2) (1984) 182–188.
[20] Y. Zhong, L. Zhang, Remote sensing image subpixel mapping based on adaptive differential evolution, IEEE Trans. Syst., Man, Cybern., Part B (Cybern.) 42 (5) (2012) 1306–1329.
[21] K. Nag, N.R. Pal, A multiobjective genetic programming-based ensemble for simultaneous feature selection and classification, IEEE Trans. Cybern. 46 (2) (2016) 499–510.
[22] H.V.H. Ayala, L. dos Santos Coelho, Tuning of pid controller based on a multiobjective genetic algorithm applied to a robotic manipulator, Expert Syst. Appl. 39 (10) (2012) 8968–8974.
[23] A.M. Mohammed, S. Li, Dynamic neural networks for kinematic redundancy resolution of parallel stewart platforms, IEEE Trans. Cybern. 46 (7) (2016) 1538–1550.
[24] X. Hu, B. Zhang, An alternative recurrent neural network for solving variational inequalities and related optimization problems, IEEE Trans. Syst., Man, Cybern., Part B (Cybern.) 39 (6) (2009) 1640–1645.
[25] Q. Liu, C. Dang, T. Huang, A one-layer recurrent neural network for real-time portfolio optimization with probability criterion, IEEE Trans. Cybern. 43 (1) (2013) 14–23.
[26] Y. Xia, G. Feng, J. Wang, A novel recurrent neural network for solving nonlinear optimization problems with inequality constraints, IEEE Trans. Neural Netw. 19 (8) (2008) 1340–1353.
[27] Q. Liu, J. Wang, A one-layer recurrent neural network with a discontinuous hard-limiting activation function for quadratic programming, IEEE Trans. Neural Netw. 19 (4) (2008) 558–570.
[28] A. Cochocki, R. Unbehauen, Neural Networks for Optimization and Signal Processing, John Wiley and Sons, Inc., 1993.
[29] C. Dang, Y. Leung, X.-b. Gao, K.-z. Chen, Neural networks for nonlinear and mixed complementarity problems and their applications, Neural Netw. 17 (2) (2004) 271–283.
[30] X. Hu, J. Wang, Solving pseudomonotone variational inequalities and pseudoconvex optimization problems using the projection neural network, IEEE Trans. Neural Netw. 17 (6) (2006) 1487–1499.
[31] X. Hu, J. Wang, A recurrent neural network for solving a class of general variational inequalities, IEEE Trans. Syst., Man, Cybern., Part B (Cybern.) 37 (3) (2007) 528–539.
[32] X.-B. Gao, Exponential stability of globally projected dynamic systems, IEEE Trans. Neural Netw. 14 (2) (2003) 426–431.
[33] S. Zhang, A. Constantinides, Lagrange programming neural networks, IEEE Trans. Circuits Syst. II: Analog Digit. Signal Process. 39 (7) (1992) 441–452.
[34] X. Zhu, S.-W. Zhang, A.G. Constantinides, Lagrange neural networks for linear programming, J. Parallel Distrib. Comput. 14 (3) (1992) 354–360.
[35] V. Sharma, R. Jha, R. Naresh, An augmented Lagrange programming optimization neural network for short term hydroelectric generation scheduling, Eng. Opt. 37 (2005) 479–497.
[36] J. Liang, H.C. So, C.S. Leung, J. Li, A. Farina, Waveform design with unit modulus and spectral shape constraints via Lagrange programming neural network, IEEE J. Sel. Top. Signal Process. 9 (8) (2015) 1377–1386.
[37] J. Liang, C.S. Leung, H.C. So, Lagrange programming neural network approach for target localization in distributed MIMO radar, IEEE Trans. Signal Process. 64 (6) (2016) 1574–1585.
[38] C.S. Leung, J. Sum, H.C. So, A.G. Constantinides, F.K. Chan, Lagrange programming neural networks for time-of-arrival-based source localization, Neural Comput. Appl. 24 (1) (2014) 109–116.
[39] R. Feng, C. Leung, A.G. Constantinides, W. Zeng, Lagrange programming neural network for nondifferentiable optimization problems in sparse approximation, IEEE Trans. Neural Netw. Learn. Syst. 28 (10) (2017) 2395–2407.
[40] M. Nagamatu, T. Yanaru, On the stability of Lagrange programming neural networks for satisfiability problems of prepositional calculus, Neurocomputing 13 (2) (1996) 119–133.
[41] C.J. Rozell, D.H. Johnson, R.G. Baraniuk, B.A. Olshausen, Sparse coding via thresholding and local competition in neural circuits, Neural Comput. 20 (10) (2008) 2526–2563.
[42] A. Balavoine, J. Romberg, C.J. Rozell, Convergence and rate analysis of neural networks for sparse approximation, IEEE Trans. Neural Netw. Learn. Syst. 23 (9) (2012) 1377–1389.
[43] D. Liu, J. Liang, A Bayesian approach to diameter estimation in the diameter control system of silicon single crystal growth, IEEE Trans. Instrum. Meas. 60 (4) (2011) 1307–1315.
[44] S.J. Ahn, W. Rauh, H.-J. Warnecke, Least-squares orthogonal distances fitting of circle, sphere, ellipse, hyperbola, and parabola, Pattern Recognit. 34 (12) (2001) 2283–2303.
[45] E.S. Maini, Enhanced direct least square fitting of ellipses, Int. J. Pattern Recognit. Artif. Intell. 20 (6) (2006) 939–953.
[46] A. Fitzgibbon, M. Pilu, R. Fisher, Direct least square fitting of ellipses, IEEE Trans. Pattern Anal. Mach. Intell. 21 (5) (1999) 476–480.
[47] A. Balavoine, C.J. Rozell, J. Romberg, Global convergence of the locally competitive algorithm, in: 2011 Digital Signal Processing and Signal Processing Education Meeting (DSP/SPE), 2011, pp. 431–436.

Zhanglei Shi is currently pursuing Ph.D. degree from the Department of Electronic Engineering, City University of Hong Kong. His current research interests include neural networks and machine learning.

Hao Wang received the PhD. degree in electronic engineering from the City University of Hong Kong in 2019. His current research interests include Neural Networks and Machine Learning.

Chi-Sing Leung received the Ph.D. degree in computer science from the Chinese University of Hong Kong, Hong Kong, in 1995. He is currently a Professor with the Department of Electrical Engineering, City University of Hong Kong, Hong Kong. He has authored over 120 journal papers in the areas of digital signal processing, neural networks, and computer graphics. His current research interests include neural computing and computer graphics.

Hing Cheung So was born in Hong Kong. He received the B.Eng. degree from the City University of Hong Kong and the Ph.D. degree from The Chinese University of Hong Kong, both in electronic engineering, in 1990 and 1995, respectively. From 1996 to 1999, he was a Research Assistant Professor with the Department of Electrical Engineering, City University of Hong Kong, where he is currently an Associate Professor. His research interests include detection and estimation, fast and adaptive algorithms, multidimensional harmonic retrieval, robust signal processing, sparse approximation, and source localization.

Junli Liang was born in China. He received the Ph.D. degree in signal and information processing from the Institute of Acoustics, Chinese Academy of Sciences. Currently, he is working as a professor at School of Electronics and Information, Northwestern Polytechnical University, China. His research interests include signal processing and image processing, and their applications.

Kim-Fung Tsang is an Associate Professor in the Department of Electrical Engineering, City University of Hong Kong. He has published more than 200 technical papers and four books/book chapter. He has been dedicated to Internet of Things (IoT) development including mobile phone infrastructure, wireless home/office/building automation and energy management system, location tracking, healthcare, smart transportation, security,.... etc. KF is now devoting his effort to IoT LPWAN development including LoRa, SigFox and NB IoT. KF is also an advisor to the government on Frequent Spectrum (OFCA), Smart Lamppost (OGCIO), the deployment of IoT infrastructure (EMSD). Dr. Tsang is currently active in the following participation and capacities: Internationally, the Chairman of IEEE Standard P2668 "IoT Index"; Chairman of NB IoT Work Group for IEEE P1451.5 smart sensors; a member of the IEEE1932.1 standard for License/Unlicensed Spectrum Interoperability in Wireless Mobile Networks Working Group; Technical Committee member of Industrial Wireless Guidelines for NIST, U.S. Department of Commerce, USA; Membership Champion of IEEE Industrial Electronics Society; Associate Editor of IEEE Transactions on Industrial Electronics; Associate Editor of IEEE Transactions on Industrial informatics; Associate Editor of IEEE Industrial Electronics Magazine.

Anthony G. Constantinides is currently the Professor of Communications and Signal Processing with Imperial College London, London, U.K. He has been actively involved in research in various aspects of digital signal processing for more than 45 years. He has authored several books and over 400 articles in digital signal processing. Prof. Constantinides is a fellow of the Royal Academy of Engineering, the Institute of Electrical and Electronics Engineers, USA, and the Institution of Electrical Engineers, U.K. He has served as the First President of the European Association for Signal Processing and has contributed in this capacity to the establishment of the European Journal for Signal Processing. He received the Medal of the Association, Palmes Academiques in 1986, and the Medal of the University of Tienjin, Shanghai, China, in 1981. He received honorary doctorates from European and Far Eastern Universities. Among these, he values highly the honorary doctorate from the National Technical University of Athens, Athens, Greece. He has organized the first international series of meetings on Digital Signal Processing, London, initially in 1967, and in Florence with Prof. V. Cappellini at the University of Florence, Florence, Italy, since 1972. In 1985, he was decorated by the French government with the Honour of Chevalier, Palmes Academiques, and in 1996 with the elevation to Officer, Palmes Academiques. His life work has been recorded in a series of audio and video interviews for the IEEE (USA) Archives as a Pioneer of Signal Processing. He has acted as an Advisor to many organizations and governments on modern technology and development. He has served on the Professorial Selection Committees around the world (15 during the last five years) and the EU University Appraising Panels, and as a member of IEE/IEEE Awards Committees and the Chair (or Co-Chair) of international conferences.