(19)【発行国】日本国特許庁(JP)
(12)【公報種別】特許公報(B2)
(11)【特許番号】5788870
(24)【登録日】2015年8月7日
(45)【発行日】2015年10月7日
(54)【発明の名称】生体リズム信号出力装置
(51)【国際特許分類】
G06F 15/02 20060101AFI20150917BHJP
A63F 9/06 20060101ALI20150917BHJP
【ＦＩ】
G06F15/02 340A
A63F9/06
【請求項の数】6
【全頁数】18
(21)【出願番号】特願2012-510578(P2012-510578)
(86)(22)【出願日】2011年4月15日
(86)【国際出願番号】JP2011002223
(87)【国際公開番号】WO2011129119
(87)【国際公開日】20111020
【審査請求日】2014年4月7日
(31)【優先権主張番号】特願2010-107684(P2010-107684)
(32)【優先日】2010年4月15日
(33)【優先権主張国】JP
(73)【特許権者】
【識別番号】505265252
【氏名又は名称】有限会社アストロハート
(74)【代理人】
【識別番号】100072604
【弁理士】
【氏名又は名称】有我 軍一郎
(72)【発明者】
【氏名】塩野 恵偉
【審査官】 三坂 敏夫
(56)【参考文献】
【文献】 特開２００９-２１１６９０（ＪＰ，Ａ）
【文献】 特開昭４８-０９０２１７（ＪＰ，Ａ）
【文献】 特開２００６-３５０９８３（ＪＰ，Ａ）
【文献】 特開２００４-０７０４９５（ＪＰ，Ａ）
(58)【調査した分野】（Int.Cl.，ＤＢ名）
Ｇ０６Ｆ １５／０２
Ａ６３Ｆ ９／０６
(57)【特許請求の範囲】
【請求項1】
ユーザの誕生地、誕生日時および所在地を表すデータを含む個人情報を表すデータを取得する個人情報取得部と、
前記ユーザによって使用される特定の機器の機器使用態様を表す履歴情報を取得する履歴情報取得部と、
前記所在地を中心とする天球図における一要素の任意の日時での位相をｎｒとし、前記誕生地を中心とする前記天球図における前記一要素の起点日時での位相をｂｒとして、位相差ΘをΘ＝ｎｒ-ｂｒによって算出し、生体リズムの周波数成分としての調波の周波数をｉとし、計算を打ち切る前記周波数の上限をＮとし、前記個人情報に基づいた前記履歴情報の周波数解析によって得られる前記調波の重み付けを表す係数をａｉとし、前記周波数解析によって得られる前記調波の位相を表す係数をｂｉとして、前記ユーザの生体リズムを表す生体リズム予測値を
【数3】
$$y = \Sigma a[i] * \sin(i * (\Theta + b[i])) / \Sigma a[i]$$

を用いて算出する生体リズム予測値算出部と、
前記生体リズム予測値が表す生体リズムに同調させた生体リズム信号を生成する生体リズム信号生成部と、
前記生体リズム信号生成部によって生成された生体リズム信号を出力する生体リズム信号出力部とを備えた生体リズム信号出力装置において、
前記生体リズム信号生成部は、前記生体リズムに対する寄与率が大きい調波から前記寄与率の累積値が予め定められた閾値以上になる調波までを抽出し、抽出した調波の中から前記生体リズムに対する寄与率が最も大きい調波の周波数の整数倍の周波数を有する調波を抽出し、抽出した調波によって前記生体リズム信号を生成することを特徴とする生体リズム信号出力装置。
【請求項2】
前記履歴情報は、前記機器が使用された日時、位置および使用態様を表す情報を含み、
前記生体リズム予測値算出部は、前記使用態様に基づいて前記履歴情報をカテゴリーに分類し、分類したカテゴリー毎に前記生体リズム予測値を算出し、
前記生体リズム信号生成部は、前記カテゴリー毎の生体リズム予測値が表す生体リズムに同調させた生体リズム信号を生成することを特徴とする請求項１に記載の生体リズム信号出力装置。
【請求項3】
前記生体リズム予測値算出部は、前記誕生日時を基準として、前記天球図上の各要素と他の各要素との位相差が０となる時刻に、前記誕生日時からの経過時間に応じて重み付けをし、時間の経過とともに前記重みを小さくした前記履歴情報を、各カテゴリーに初期値として設定することを特徴とする請求項２に記載の生体リズム信号出力装置。
【請求項4】
前記生体リズム信号出力部は、前記生体リズム信号を出力するときに前記生体リズム予測値が最大となるカテゴリーの生体リズム信号を出力することを特徴とする請求項２に記載の生体リズム信号出力装置。
【請求項5】
請求項１ないし請求項４のいずれかに記載の生体リズム信号出力装置としてコンピュータを機能させるための生体リズム信号出力プログラム。
【請求項6】
請求項５に記載の生体リズム信号出力プログラムを記録したコンピュータで読み取り可能な記録媒体。

サンプルコード
---
test_biowave.py

```
# -*- coding: utf-8 -*-
"""
@author: Shiono
"""

import math
import matplotlib.pyplot as plt
```

```
import biowave


class TestDataSet:
    def dataset01(self, n, th=0):
        an = [1.0 / (i+1)  for i in range(n)]
        bn = [th for i in range(n)]
        return an, bn

    def dataset02(self, n, th=0):
        an = [1.0 / (i+1) if (i+1) % 2 == 1 else 0 for i in range(n)]
        bn = [th for i in range(n)]
        return an, bn

    def dataset03(self, n, th=0):
        an = [1.0 / ((i+1)*(i+1)) if (i+1) % 2 == 1 else 0 for i in range(n)]
        bn = [th for i in range(n)]
        return an, bn

    def dataset04(self, n, th=0):
        an = [1.0 / (i+1) if i == 0 or (i+1) % 2 == 0 else 0 for i in range(n)]
        bn = [th for i in range(n)]
        return an, bn

    def dataset05(self, n, th=0):
        an = [1.0 / (i+1)  if math.log2((i+1)).is_integer() == True else 0 for i in range(n)]
        bn = [th for i in range(n)]
        return an, bn


def ab_output(bdata):
    print(bdata)
    for i in range(len(bdata[0])):
         print("{:>2d}".format(i), "] {:>2f}".format(bdata[0][i]), " {:>2f}".format(bdata[1][i]))

def th_output(thi):
    print(thi)
    #for i in range(len(thi)):
         #print("{:>2d}".format(i), "] {:>2f}".format(thi[i]))
    fig, ax = plt.subplots()
    div = 360 / len(thi)
    x = [i * div for i in range(len(thi))]
    ax.plot(x, thi)
    plt.show()

def sublop(bdata, div=12):
    wav = biowave.BioWave()
    nn = wav.chack_data(bdata)
    if nn == 0:
        errorlog.errorlog("array size error.")
        return 0
    else:
        return wav.theta_lop(div, bdata[0], bdata[1], nn)

def subtest(n=10, th=0, div=72):
    tda = TestDataSet()
    bdata = tda.dataset01(n, th)
    ab_output(bdata)
    thi = sublop(bdata, div)
    th_output(thi)

    bdata = tda.dataset02(n, th)
    ab_output(bdata)
    thi = sublop(bdata, div)
    th_output(thi)

    bdata = tda.dataset03(n, th)
    ab_output(bdata)
    thi = sublop(bdata, div)
    th_output(thi)

    bdata = tda.dataset04(n, th)
    ab_output(bdata)
    thi = sublop(bdata, div)
    th_output(thi)

    bdata = tda.dataset05(n, th)
    ab_output(bdata)
    thi = sublop(bdata, div)
    th_output(thi)


def test_main(n=10, th=0, div=72):
    subtest(n, th, div)

print("start")
test_main(64, 0, 120)
print("end")
```

---

biowave.py

```
# -*- coding: utf-8 -*-
"""
@author: Shiono
"""

import math
import errorlog

class BioWave:
    # def __init__(self):

    def _chack_data(self, bdata):
        nn = 0
        if len(bdata[0]) == len(bdata[1]):
            nn = len(bdata[0])
        return nn

    def c_wave(self, th, an, bn, nn):
        y = 0
        for i in range(nn):
            y += an[i] * math.sin(math.radians((i + 1) * (th + bn[i])))
        return y / math.fsum(an)

    def theta_lop(self, div, an, bn, nn):
        thi = [0.0] * div
        deg = 360 / div
        theta = 0
        for i in range(div):
            thi[i] = self.c_wave(theta, an, bn, nn)
            theta += deg
        return thi


if __name__ == '__main__':
    print("not main")
```

---
errorlog.py

```
# -*- coding: utf-8 -*-
"""
@author: Shiono
"""

def errorlog(str):
    print(str)
```

フォルダ構造
```
カレント---+biowave_cy_pure---+biowave_cy_pure.pyx
                              +biowave_cy_pure_setup.py

        ---+biowave_cy  ------+

        ---+test_viowave.py
```

実行方法
カレント
```
>cd biowave_cy_pure/
>python biowave_cy_setup.py build_ext
>cp ./build/lib.win-amd64-3.7/biowave_cy_pure.cp37-win_amd64.pxd biowave_cy_pure.pxd
>cd ../
>python3 test_biowave.py
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝
```
# -*- coding: utf-8 -*-
"""
@author: Shiono
biowave_cy_pure.pyx
"""
import cython
import math
import errorlog
import numpy as np

@cython.cclass
class BioWave:

    # @cython.locals(b=cython.int)
    # def __init__(self, b):

    @cython.ccall
    @cython.returns(cython.int)
    @cython.locals(nn=cython.int)
    def chack_data(self, bdata):
```

```
        nn = len(bdata[0])
        if nn != len(bdata[1]):
            nn = 0
        return nn

    @cython.ccall
    @cython.locals(th=cython.double)
    @cython.locals(an=cython.double[:])
    @cython.locals(bn=cython.double[:])
    @cython.locals(nn=cython.int)
    @cython.returns(cython.double)
    @cython.locals(y=cython.double)
    @cython.locals(i=cython.int)
    def c_wave(self, th, an, bn, nn):
        y = 0.0
        for i in range(nn):
            y += an[i] * math.sin(math.radians((i + 1) * (th + bn[i])))
        return y / math.fsum(an)

    @cython.ccall
    @cython.locals(div=cython.int)
    @cython.locals(an=cython.double[:])
    @cython.locals(bn=cython.double[:])
    @cython.locals(nn=cython.int)
    @cython.returns(cython.double[:])
    @cython.locals(deg=cython.double)
    @cython.locals(theta=cython.double)
    @cython.locals(i=cython.int)
    @cython.locals(thi=cython.double[:])
    def theta_lop_cy(self, thi, div, an, bn, nn):
        deg = 360.0 / div
        theta = 0.0
        for i in range(div):
            thi[i] = self.c_wave(theta, an, bn, nn)
            theta += deg
        return thi

    #@cython.ccall
    @cython.locals(div=cython.int)
    @cython.locals(an=cython.double[:])
    @cython.locals(bn=cython.double[:])
    @cython.locals(nn=cython.int)
    @cython.locals(theat=cython.double[:])
    def theta_lop(self, div, an, bn, nn):
        thi = np.array([0.0 for i in range(div)])
        # print("thi=", thi, "an=", an, " bn=", bn)
        self.theta_lop_cy(thi, div, an, bn, nn)
        return thi

if __name__ == '__main__':
    print("not main")


------------------------------------
====================
====================
------------------------------------
# -*- coding: utf-8 -*-
"""
@author: Shiono
biowave_cy_setup.py
"""

# python biowave_cy_setup.py build_ext
# cp ./build/lib.win-amd64-3.7/biowave_cy_pure.cp37-win_amd64.pxd biowave_cy_pure.pxd

from distutils.core import setup
from distutils.extension import Extension
from Cython.Distutils import build_ext

import numpy as np

sourcefiles = ['biowave_cy_pure.pyx']
setup(
    cmdclass = {'build_ext':build_ext},
    ext_modules = [Extension('biowave_cy_pure', sourcefiles)],
    include_dirs = [np.get_include()]
)

------------------------------------
====================
====================
------------------------------------
★2nd sample
---------------------------------------
test_py_sample.py
---------------------------------------
# -*- coding: sjis -*-
"""
@author: Shiono
```

```
"""

import copy
import random

import py_sample as samp
import user_def as udef

class DataSet:
    def dataset01(self):
        dum = [
            [1, 1, 0, 0],
            [0, 1, 0, 0],
            [1, 1, 0, 1],
            [0, 0, 1, 1],
            [1, 1, 0, 1]
        ]
        return dum

    def dataset02(self):
        dum = [
            [0, 1, 1, 1, 1, 0],
            [1, 0, 1, 0, 0, 1],
            [1, 1, 1, 0, 0, 1],
            [0, 0, 0, 0, 1, 1],
            [0, 1, 0, 0, 0, 1],
            [0, 1, 1, 0, 1, 0],
        ]
        return dum

    def dataset03(self):
        dum = [
            [0, 1, 1, 1, 1, 0,0,1],
            [1, 0, 1, 0, 0, 1,0,0],
            [1, 1, 1, 0, 0, 0,1,1],
            [0, 0, 0, 0, 1, 1,0,0],
            [0, 1, 0, 0, 0, 1,1,1],
            [0, 1, 1, 0, 1, 0,1,0],
            [0, 1, 1, 0, 1, 0,1,0],
            [0, 1, 1, 0, 1, 0,1,0],
        ]
        return dum

    def dataset04(self):
        dum = [
            [1, 1, 1, 1, 1, 1,1,1,1],
            [1, 0, 0, 0, 1, 0,0,0,1],
            [1, 0, 1, 0, 1, 0,0,0,1],
            [1, 0, 1, 0, 1, 0,0,0,1],
            [1, 0, 1, 0, 1, 0,1,0,1],
            [1, 0, 0, 0, 1, 0,1,0,1],
            [1, 1, 1, 1, 1, 0,1,0,1],
            [1, 0, 0, 0, 1, 0,1,0,1],
            [1, 0, 1, 0, 1, 0,0,1,1],
            [1, 0, 0, 0, 1, 0,0,1,1],
            [1, 0, 0, 0, 1, 0,0,1,1],
            [1, 1, 1, 1, 1, 1,1,1,1],
        ]
        return dum

    def dataset05(self, startnum=10, stopnum=100, stepnum=10):
        x = random.randrange(startnum, stopnum, stepnum)
        y = random.randrange(startnum, stopnum, stepnum)
        dum = [[0 for i in range(x)] for j in range(y)]
        for i in range(y):
            for j in range(x):
                dum[i][j] = random.randint(0, 1)
        return dum


def subtest(dim, x, y, mc, md, bx, by, deb=1):
    la = samp.Land(x, y, mc, md)
    la.set_dim(dim, x, y)
    dum = copy.deepcopy(dim)
    a = 0
    for i in range(1, y + 1):
        for j in range(1, x + 1):
            if la.isdim(j, i) == 1:
                a = a + 1
                for k in range(mc + 1):
                    bx[k] = j; by[k] = i
                res = la.c_land2(bx, by, 0)

                if deb > 0 and res == 0:
                    la.get_dim(dum, x, y)
    print("{:>2d}".format(a))

def test_main(c, startnum=10, stopnum=100):
    ds = DataSet()
    if c == 0:
```

```
        dim = ds.dataset01()
    elif c == 1:
        dim = ds.dataset02()
    elif c == 2:
        dim = ds.dataset03()
    elif c == 3:
        dim = ds.dataset04()
    else:
        dim = ds.dataset05(startnum=startnum, stopnum=stopnum)

    #print(dim)
    x = len(dim[0])
    y = len(dim)
    mc = udef.MAX_C
    md = udef.MAX_D
    bx = [0] * (udef.MAX_C + 1)
    by = [0] * (udef.MAX_C + 1)
    subtest(dim, x, y, mc, md, bx, by)

test_main(0)
print("")
test_main(1)
print("")
test_main(2)
print("")
test_main(3)
print("")
test_main(4, 10, 100)
print("4-1")
#test_main(4, 10, 200)
#print("4-2")
#test_main(4, 10, 500)


----------------------------------------
py_sample.py
----------------------------------------
# -*- coding: utf-8 -*-
"""
@author: Shiono
"""

import user_def as udef

class Land():
    def __init__(self, max_x=udef.MAX_X, max_y=udef.MAX_Y, max_c=udef.MAX_C, max_d=udef.MAX_D):
        self.dim = [[0 for x in range(max_x + 2)] for y in range(max_y + 2)]
        self.xx = [0] * (max_c + 1)
        self.yy = [0] * (max_c + 1)
        self.rxx = [0] * (max_d + 1)
        self.ryy = [0] * (max_d + 1)
        self.max_x = max_x
        self.max_y = max_y
        self.max_c = max_c
        self.max_d = max_d

    def isdim(self, x, y):
        return self.dim[y][x]

    def c_land(self, x, y, c):
        if self.dim[y][x] != 1: return 0
        else:
            self.xx[c] = x
            self.yy[c] = y
            if c >= self.max_c: return -1
            else:
                self.dim[y][x] = 3
                if self.c_land(x + 1, y, c + 1) < 0: return -1
                if self.c_land(x - 1, y, c + 1) < 0: return -1
                if self.c_land(x, y + 1, c + 1) < 0: return -1
                if self.c_land(x, y - 1, c + 1) < 0: return -1
                return 0

    def c_land2(self, x1, y1, c):
        self.rxx[c] = x1
        self.ryy[c] = y1
        if c >= self.max_d: return -1
        else:
            for i in range(self.max_c + 1):
                x2 = x1[i]
                y2 = y1[i]
                self.dim[y2][x2] = 1
                if self.c_land(x2, y2, 0) == 0:
                    continue
                else:
                    if self.c_land2(self.xx, self.yy, c + 1) < 0: return -1

            return 0

    def c_land3(self, x1, y1, c):
```

```
        for i in range(self.max_d + 1):
            if c_land2(x1[i], y1[i], 0) == 0: continue
            else:
                c_land3(self.rxx, self.ryy, c + 1)
        return 0

    def set_dim(self, dum, mx, my):
        mx1 = mx + 1
        my1 = my + 1
        for i in range(my1 + 1):
            for j in range(mx1 + 1):
                if (i == 0 or i == my1 or j == 0 or j == mx1):
                    self.dim[i][j] = 5
                else:
                    self.dim[i][j] = dum[i - 1][j - 1]

    def get_dim(self, dum, mx, my):
        for i in range(1, my + 1):
            for j in range(1, mx + 1):
                dum[i - 1][j - 1] = self.dim[i][j]
                print("{:>2d}".format(self.dim[i][j]), end='')
            print("")
        print("")


if __name__ == '__main__':
    print('not main')
```

---

user_def.py

```
# -*- coding: utf-8 -*-
"""
@author: Shiono
"""

MAX_Y = 1000
MAX_X = 1000
MAX_C = 800
MAX_D = 800
```

実行方法

```
>python3 test_py_sample.py
```

---

json_sample.json

```
sample_data_sets =
    [
        {
            "set_id":"xxx",
            "set_name":"XXXX",
            "date":"20220125",
            "hhmmss":"091510",
            "result_value":1234,
            "movies":{
                 "dir":["",,,,],
                 "flname":[["AAA1","AAA2",,,,],,,],
                 "feature":{
                     "key":[{"key_name":[],"key_value":[[],[],,,,]},{},,,],
                     "result":[res1,res2,,,,],
                     "weight":[var1,var2,,,,]
                 }
            },
            "gps":{
                 "dir":"",
                 "flname":[["BBB1","BBB2",,,,],,,],
                 "feature":{
                     "key":[{"key_name":[],"key_value":[[],[],,,,]},{},,,],
                     "result":[res1,res2,,,,],
                     "weight":[var1,var2,,,,]
                 }
            },
            "sound":{
                 "dir":["",,,,],
                 "flname":[["CCC1","CCC2",,,,],,,],
                 "feature":{
                     "key":[{"key_name":[],"key_value":[[],[],,,,]},{},,,],
                     "result":[res1,res2,,,,],
                     "weight":[var1,var2,,,,]
                 }
            },
            "text":{
                 "dir":["",,,,],
                 "flname":[["DDD1","DDD2",,,,],,,],
```

```
                "feature":{
                    "key":[{"key_name":[],"key_value":[[],[],,,,]},{},,,],
                    "result":[res1,res2,,,,],
                    "weight":[var1,var2,,,,]
                }
            }
        }
    ]

====================================
END
====================================
```