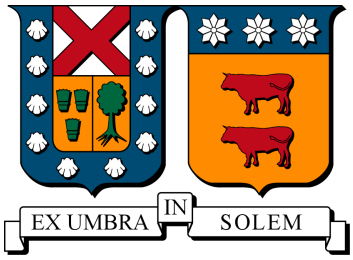
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAÍSO - CHILE



"SISTEMA DE ESTIMULACIÓN VISUAL Y REGISTRO DE MOVIMIENTOS OCULARES PARA TAREAS SICOMOTORAS"

CHRISTIAN ANDRÉS WICHE LATORRE

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELECTRÓNICO MENCIÓN CONTROL E INSTRUMENTACIÓN

PROFESOR GUÍA: **MARÍA JOSÉ ESCOBAR**
PROFESOR CORREFERENTE: **MATÍAS ZAÑARTU**

?? - 2018

Inserte su dedicatoria aquí

Firma

Agradecimientos

Resumen

Abstract

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Motivación

- frase inicial:

1.2. Objetivos

- Sistema

Capítulo 2

Estado del arte

2.1. Sistemas de seguimiento ocular

2.1.1. Movimiento ocular

2.1.2. Métodos de captura

Un poco de historia

Tecnologías actuales

1. **De contacto directo**
2. **Seguimiento ocular**
3. **Medición de potencial eléctrico**

Comparativa

2.1.3. Sistemas comerciales más relevantes

1. **EyeGaze**
2. **EyeLink**
3. **EyeTribe**

4. **IViewX**

5. **Tobii**

2.2. Sistemas de estimulación visual

2.2.1. Hardware de estimulación

Un poco de historia

Tecnologías actuales

1. **Monitores CRT**

2. **Monitores LED, oLED, LCD**

Comparativa

2.2.2. Software de estimulación

Software más relevante

1. **PsychoPy**

2. **PsychoToolbox**

3. **VissionEgg**

4. **Presentation**

Comparativa

2.2.3. Experimentos de estimulación

Capítulo 3

Sistema propuesto

3.1. Alcances

3.1.1. Utilidad de la GUI

3.1.2. Experimentos a implementar

3.2. Tecnologías utilizadas

3.2.1. Hardware

3.2.2. Software

3.3. Diseño e implementación de la GUI

3.3.1. Arquitectura

3.3.2. Archivos de configuración

3.3.3. Script de ejecución

Capítulo 4

Resultados

4.1. Configuración de test de prueba

4.2. Mediciones obtenidas

Capítulo 5

Conclusiones y trabajo futuro

5.1. Conclusiones

5.2. Trabajo futuro

Referencias

[1] I. Steinecke and H. Herzel, "Bifurcations in an asymmetric vocal - fold model," *The Journal of the Acoustical Society of America*, vol. 97, pp. 1874 – 1884, 1995.

[2] K. Honda, "Physiological processes of speech production," *Handbook of Speech Processing, Springer*, pp. 7–26, 2008.

[3] E. Simoncelli and B. Olshausen, "Natural image statistics and neural representation," *Annual Review in Neuroscience*, vol. 24, pp. 1193–1216, 2001.

[4] N. Lesica, T. Ishii, G. Stanley, and T. Hosoya, "Estimating receptive fields from responses to natural stimuli with asymmetric intensity distributions," *PLoS ONE*, vol. 3(8): e3060, 2008.

[5] J. Touryan, G. Felsen, and Y. Dan, "Spatial structure of complex cell receptive fields measured with natural images," *Neuron 45*, vol. 45(5), pp. 781–791, 2005.

[6] T. Sharpee, N. Rust, and W. Bialek, "Analyzing neural responses to natural signals: Maximally informative dimensions," *Neural Computation*, vol. 16(2), pp. 223–250, 2004.

[7] G. Rousselet, S. Thorpe, and M. Fabre-Thorpe, "How parallel is visual processing in the ventral pathway?" *Trends in Cognitive Sciences*, vol. 8(8), pp. 363–370, 2004.

[8] R. Guyonneau, R. VanRullen, and S. Thorpe, "Temporal codes and sparse representations: A key to understanding rapid processing in the visual system," *Journal of Physiology-Paris*, vol. 98(4-6), pp. 487 – 497, 2004.

[9] E. D. Young, "Physiological acoustics," *Handbook of Acoustics, Springer*, pp. 429 – 457, 2007.

[10] F. Jaramillo, V. Markin, and A. Hudspeth, "Auditory illusions and the single hair cell," *Nature*, vol. 364, pp. 527 – 529, 1993.

[11] T. Gollisch, "Rapid neural coding in the retina with relative spike latencies," *Science*, vol. 319, pp. 1108 – 2008, 2008.

Apéndice A

Código fuente

```
%Programa Principal
tstart = 0; tend = 20; dt = 0.01;
data = simulacion(tstart, tend, dt)
visualize_test(data);
Graficos(data);
```

```
# -*- coding: utf-8 -*-
# ============================================================================
# Modules
# ============================================================================
import os
import platform as pt
from PyQt4 import QtGui
import ConfigParser as cp

# ============================================================================
# ID Definitions
# ============================================================================
TEST_TYPE = u'.tst'
EXPS_TYPE = u'.exp'
# =================
NOT_VALID_CHARS = [u'', u'.', u'..']


# ============================================================================
# Config Files Handler
# ============================================================================
class confHandler(cp.ConfigParser):
    def __init__(self):
        cp.ConfigParser.__init__(self)

    # ================================ File Handler
    def openConf(self, fileName):
        fileName = unicode(fileName)
        if os.path.isfile(fileName):
            fileItem = open(fileName)
            self.readfp(fileItem)
            fileItem.close()
            return True
        else:
            return False

```

```
37      def saveConf(self, fileName, isRW=True):
38          fileName = unicode(fileName)
39          isFile = os.path.isfile(fileName)
40          if not isFile or (isFile and isRW):
41              fileItem = open(fileName, 'w')
42              self.write(fileItem)
43              fileItem.close()
44              return True
45          else:
46              return False
47
48      # ================================ Data Handler
49      def getSections(self):
50          sections = self.sections()
51          if sections:
52              return sections
53          else:
54              return []
55
56      def getValue(self, section, option, mode=None):
57          if mode is int:
58              return self.getint(section, option)
59          elif mode is float:
60              return self.getfloat(section, option)
61          elif mode is bool:
62              return self.getboolean(section, option)
63          else:
64              return self.get(section, option)
65
66      def setValue(self, section, option, value):
67          try:
68              sections = self.getSections()
69              if section in sections:
70                  self.set(section, option, value)
71              else:
72                  self.add_section(section)
73                  self.set(section, option, value)
74              return True
75          except:
76              return False
77
78      # ================================ Children
79      def getConf(self, fileName=u''):
80          pass
81
82      def putConf(self, fileName=u''):
83          pass
84
85
86  # ============================================================================
87  # Config File Class
88  # ============================================================================
89  class confFile(confHandler):
90      def __init__(self):
91          confHandler.__init__(self)
92          # ==============
93          self.fileName = u'./saccadeApp.ini'
94          # ==============
95          if pt.system() is 'Windows':
96              self.secConf = u'ConfigWindows'
97          else:
98              self.secConf = u'ConfigUnix'
99          # ==============
100         self.testDir = u''
101         self.expsDir = u''
102         # ==============
103         self.getConfig()
104
```

```
105     # ================================
106     def getConfig(self, filename=u''):
107         # ==============
108         if self.openConf(fileName=self.fileName) and self.secConf in self.getSections():
109             self.testDir = self.getValue(self.secConf, u'testDir', mode=str)
110             self.expsDir = self.getValue(self.secConf, u'expsDir', mode=str)
111         else:
112             self.testDir = u'Tests/'
113             self.expsDir = u'Experiments/'
114             self.putConfig()
115         # ==============
116         if not os.path.isdir(self.testDir):
117             os.mkdir(self.testDir)
118         if not os.path.isdir(self.expsDir):
119             os.mkdir(self.expsDir)
120 
121     def putConfig(self, filename=u''):
122         self.setValue(self.secConf, u'testDir', self.testDir)
123         self.setValue(self.secConf, u'expsDir', self.expsDir)
124         # ==============
125         self.saveConf(fileName=self.fileName, isRW=True)
126 
127 
128 # ============================================================================
129 # Test File Class
130 # ============================================================================
131 class testFileConf(confHandler):
132     def __init__(self, fileName=u''):
133         confHandler.__init__(self)
134         # ==============
135         self.fileName = fileName
136         # ==============
137         self.imagData = []
138         self.extraFlag = False
139         self.extraImag = u''
140         self.extraKeys = []
141         self.extraCAns = u''
142         # ==============
143         if self.fileName is not u'':
144             self.getConf()
145 
146     # ================================
147     def getConf(self, fileName=u''):
148         # ==============
149         if fileName is not u'':
150             self.fileName = fileName
151         # ==============
152         if self.fileName is not u'' and self.openConf(fileName=self.fileName):
153             self.imagData = self.getImagData()
154             self.extraFlag = self.getValue(u'finalSel', u'extraFlag', mode=bool)
155             self.extraImag = self.getValue(u'finalSel', u'extraImag', mode=str)
156             self.extraKeys = self.getExtraKeys()
157             self.extraCAns = self.getValue(u'finalSel', u'extraCAns', mode=str)
158             return True
159         else:
160             return False
161 
162     def putConf(self, fileName=u''):
163         # ==============
164         if fileName is not u'':
165             self.fileName = fileName
166         # ==============
167         self.setValue(u'mainTest', u'imagData', self.setImagData())
168         self.setValue(u'finalSel', u'extraFlag', self.extraFlag)
169         self.setValue(u'finalSel', u'extraImag', self.extraImag)
170         self.setValue(u'finalSel', u'extraKeys', self.setExtraKeys())
171         self.setValue(u'finalSel', u'extraCAns', self.extraCAns)
172         # ==============
```

```
173         return self.saveConf(fileName=self.fileName, isRW=True)
174
175     # ================================
176     def setImagData(self):
177         if self.imagData:
178             auxStr = u''
179             for line in self.imagData:
180                 auxStr += u'\n%s\t%s' % tuple([line[0], line[1]])
181             return auxStr
182         else:
183             return u''
184
185     def setExtraKeys(self):
186         if self.extraKeys:
187             auxStr = u''
188             for key in self.extraKeys:
189                 auxStr += u' %s;' % key
190             return auxStr
191         else:
192             return u''
193
194     # ================================
195     def getImagData(self):
196         auxStr = self.getValue(u'mainTest', u'imagData', mode=str)
197         if auxStr is not u'':
198             auxOut = []
199             auxArr = auxStr.split(u'\n')
200             for line in auxArr:
201                 try:
202                     auxLne = line.split(u'\t')
203                     auxOut.append([auxLne[0], float(auxLne[1])])
204                 except:
205                     pass
206             return auxOut
207         else:
208             return []
209
210     def getExtraKeys(self):
211         auxStr = self.getValue(u'finalSel', u'extraKeys', mode=str)
212         if auxStr is not u'':
213             auxOut = []
214             auxArr = auxStr.split(u';')
215             for line in auxArr:
216                 if line is not u'':
217                     auxOut.append(line)
218             return auxOut
219         else:
220             return []
221
222
223 # ============================================================================
224 # Experiment File Class
225 # ============================================================================
226 class expsFileConf(confHandler):
227     def __init__(self, fileName=u''):
228         confHandler.__init__(self)
229         # ==============
230         self.fileName = fileName
231         # ==============
232         self.testData = []
233         self.randFlag = False
234         self.restFlag = False
235         self.restTest = 0
236         self.restTime = 0.0
237         # ==============
238         if self.fileName is not u'':
239             self.getConf()
240
```

```
241     # ================================
242     def getConf(self, fileName=u''):
243         # ==============
244         if fileName is not u'':
245             self.fileName = fileName
246         # ==============
247         if self.fileName is not u'' and self.openConf(fileName=self.fileName):
248             self.testData = self.getTestData()
249             self.randFlag = self.getValue(u'testSort', u'randFlag', mode=bool)
250             self.restFlag = self.getValue(u'testRest', u'restFlag', mode=bool)
251             self.restTest = self.getValue(u'testRest', u'restTest', mode=int)
252             self.restTime = self.getValue(u'testRest', u'restTime', mode=float)
253             return True
254         else:
255             return False
256 
257     def putConf(self, fileName=u''):
258         # ==============
259         if fileName is not u'':
260             self.fileName = fileName
261         # ==============
262         self.setValue(u'mainExps', u'testData', self.setTestData())
263         self.setValue(u'testSort', u'randFlag', self.randFlag)
264         self.setValue(u'testRest', u'restFlag', self.restFlag)
265         self.setValue(u'testRest', u'restTest', self.restTest)
266         self.setValue(u'testRest', u'restTime', self.restTime)
267         # ==============
268         return self.saveConf(fileName=self.fileName, isRW=True)
269 
270     # ================================
271     def setTestData(self):
272         if self.testData:
273             auxStr = u''
274             for line in self.testData:
275                 auxStr += u'\n%s\t%s' % tuple([line[0], line[1]])
276             return auxStr
277         else:
278             return u''
279 
280     # ================================
281     def getTestData(self):
282         auxStr = self.getValue(u'mainExps', u'testData', mode=str)
283         if auxStr is not u'':
284             auxOut = []
285             auxArr = auxStr.split(u'\n')
286             for line in auxArr:
287                 try:
288                     auxLne = line.split(u'\t')
289                     auxOut.append([auxLne[0], float(auxLne[1])])
290                 except:
291                     pass
292             return auxOut
293         else:
294             return []
295 
296 
297 # ============================================================================
298 # Experiment File Class
299 # ============================================================================
300 def getFilePath(path, ext, newName=u'', oldName=u'', isRW=False):
301     newName = unicode(newName) if newName is not None else u''
302     oldName = unicode(oldName) if oldName is not None else u''
303     auxName = oldName
304     newPath = None
305     # ==============
306     isFirst = True
307     isReady = False
308     # ==============
```

```
309     while not isReady:
310         if isFirst and newName != u'':
311             isFirst = False
312             isOk = True
313         else:
314             isFirst = False
315             newName, isOk = QtGui.QInputDialog.getText(None, u'New file name', u'Insert new
    file name:', text=auxName)
316             newName = unicode(newName)
317         # ==============
318         if isOk:
319             newPath = unicode(path + u'/' + newName + ext)
320             isNewPathOk, isNewPathExist = isPathAvailable(newPath)
321             # ==============
322             if isNewPathOk and newName not in NOT_VALID_CHARS and len(newName) >= 4:
323                 if isRW and isNewPathExist and newName != oldName:
324                     print u'Error: This file name affects not-involved files.'
325                     QtGui.QMessageBox.about(None, u'Error!', u'This file name affects\nnot-
    involved files')
326                 elif not isRW and isNewPathExist:
327                     print u'Error: Cannot copy to a existing file.'
328                     QtGui.QMessageBox.about(None, u'Error!', u'Cannot copy to a existing
    file')
329                 else:
330                     isReady = True
331             else:
332                 if len(newName) < 4:
333                     print u'Error: file name too short'
334                     QtGui.QMessageBox.about(None, u'Error!', u'Please use a name larger\
    nthan 4 letters.')
335                 else:
336                     print u'Error: Bad file name'
337                     QtGui.QMessageBox.about(None, u'Error!', u'Bad file name')
338             # ==============
339             auxName = newName
340         else:
341             print u'Operation cancelled.'
342             newName = None
343             newPath = None
344             isReady = True
345     # ==============
346     return newName, newPath
347
348
349 def isPathAvailable(path):
350     # Return: [nameOk, exists]
351     if not os.path.isfile(path):
352         try:
353             auxFile = open(path, 'w')
354             auxFile.close()
355             os.remove(path)
356             return True, False
357         except:
358             return False, False
359     else:
360         return True, True
361     pass
```

Apéndice B

Vistas de la GUI