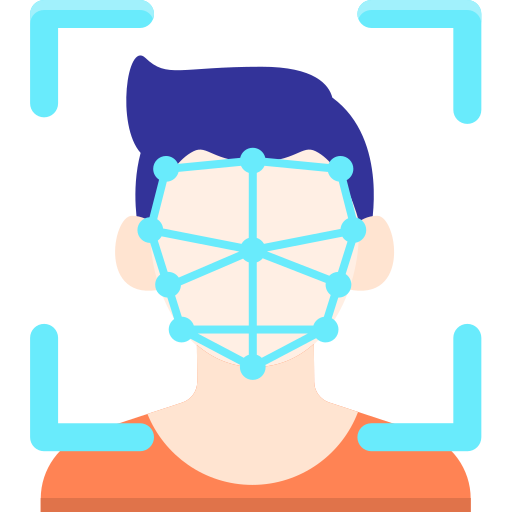
**PROJETO FINAL**

LABORATÓRIO DE INOVAÇÃO E AUTOMAÇÃO I



**Controle de Acesso com Reconhecimento Facial**

**Autores:** Bernardo Halfeld de Assis Nébias (202004705)

Luiz Henrique Keng Queiroz Júnior (202000623)

**Descrição:** Este é um projeto de reconhecimento facial para controle de acesso em ambientes, onde foi utilizada a linguagem Python, juntamente com a biblioteca “Face Recognition” para a etapa de reconhecimento, e para a parte física do projeto foi utilizada a plataforma Arduino, em conjunto com a biblioteca “Pyfirmata” para o controle dos componentes conectados a placa, via Python.

**Passos Envolvidos:**

1. **Preparação dos dados e utilização do modelo:** É necessário realizar o cadastro de um novo usuário, através do código “Registro.py”. Nele o usuário captura uma imagem do seu rosto que será salva com o nome colocado pelo mesmo. A imagem é salva em uma pasta local que servirá como database para o processo de reconhecimento. Para o reconhecimento é utilizada a biblioteca “face\_recognition”, onde em um intervalo de 5 segundos é capturada uma imagem do usuário. Essa imagem passará pelo método “recognize”, onde será comparada com as imagens contidas no Database. Caso o usuário seja reconhecido, é liberado o acesso do mesmo, senão, o processo é mantido.
2. **Funcionamento do modelo:** A biblioteca “face\_recognition” possui modelos e métodos pré-treinados e desenvolvidos utilizando técnicas de deep learning com precisão de até 99,38%. Isso nos poupa o trabalho de construção e treinamento de um modelo próprio, que dificilmente seria tão eficiente. Através do método “face\_encondings” as imagens cadastradas são convertidas em codificações numéricas (parâmetros) e o método “compare\_faces” compara esses parâmetros entre as imagens para verificar se há correspondência. No nosso caso, a comparação é feita entre a imagem capturada no ciclo de 5 segundos e as imagens contidas no database do projeto. Caso seja obtida alguma correspondência, é retornado o título da imagem contida no database, que no presente projeto seria o nome do usuário em questão
3. **Implementação do sistema:** No funcionamento do sistema, o reconhecimento é realizado sempre que a porta está fechada, detecção feita pelo sensor Reed Switch. Caso o usuário seja reconhecido pelos processos descritos acima, o servo motor destrava a tranca da porta e o led verde é ativado, informando que o acesso foi liberado. Quando a porta for fechada novamente, a tranca será travada e o processo de reconhecimento rodará novamente.
4. **Perspectivas:** Como perspectiva de melhoria a ser implementada está o filtro anti-spoofing para prevenir acesso não-autorizado através do uso de fotos ou vídeos.

**Códigos do Projeto:**

* Código “utility.py”, utilizado como uma “classe” de funções suporte, que serão utilizadas nos códigos principais repetidas vezes.
* import os
* import pickle
* import tkinter as tk
* from tkinter import messagebox
* import face\_recognition
* import sys
* import time
* #from pyfirmata import Arduino, util
* from time import sleep
* #PORT = 'COM3'
* #board = Arduino(PORT)
* def get\_button(window, text, color, command, fg='white'):
* button = tk.Button(
* window,
* text=text,
* activebackground="black",
* activeforeground="white",
* fg=fg,
* bg=color,
* command=command,
* height=2,
* width=20,
* font=('Helvetica bold', 20)
* )
* return button
* def get\_small\_button(window, text, color, command, fg='white'):
* button = tk.Button(
* window,
* text=text,
* activebackground="black",
* activeforeground="white",
* fg=fg,
* bg=color,
* command=command,
* height=1,
* width=20,
* font=('Helvetica bold', 20)
* )
* return button
* def get\_img\_label(window):
* label = tk.Label(window)
* label.grid(row=0, column=0)
* return label
* def get\_text\_label(window, text):
* label = tk.Label(window, text=text)
* label.config(font=("sans-serif", 21), justify="left")
* return label
* def get\_entry\_text(window):
* inputtxt = tk.Text(window,
* height=2,
* width=15, font=("Arial", 32))
* return inputtxt
* def msg\_box(title, description):
* messagebox.showinfo(title, description)
* def recognize(img, db\_path):
* # it is assumed there will be at most 1 match in the db
* embeddings\_unknown = face\_recognition.face\_encodings(img)
* if len(embeddings\_unknown) == 0:
* return 'no\_persons\_found'
* else:
* embeddings\_unknown = embeddings\_unknown[0]
* db\_dir = sorted(os.listdir(db\_path))
* match = False
* j = 0
* while not match and j < len(db\_dir):
* path\_ = os.path.join(db\_path, db\_dir[j])
* file = open(path\_, 'rb')
* embeddings = pickle.load(file)
* match = face\_recognition.compare\_faces([embeddings], embeddings\_unknown)[0]
* j += 1
* if match:
* return db\_dir[j - 1][:-7]
* else:
* return 'unknown\_person'
* Código “Registro.py”, responsável pela interface de cadastro de usuários na aplicação.
* import os.path
* import pickle
* from tkinter import \*
* from tkinter import font
* from tkinter import messagebox
* import tkinter as tk
* import cv2
* from PIL import Image, ImageTk
* import face\_recognition
* import utility
* db\_dir = './db'
* class App:
* def \_\_init\_\_(self):
* self.main\_window = tk.Tk()
* self.main\_window.geometry("1200x620+350+100")
* self.main\_window.attributes('-fullscreen', True)
* self.main\_window.overrideredirect(False)
* self.text\_label\_main\_window\_title = tk.Label(self.main\_window, text = 'Reconhecimento Facial', font=("Arial", 30, "bold"))
* self.text\_label\_main\_window\_title.place(x = 700, y = 100)
* self.webcam\_label = utility.get\_img\_label(self.main\_window)
* self.webcam\_label.place(x=400, y=250, width=700, height=500)
* self.add\_webcam(self.webcam\_label)
* self.process\_webcam()
* self.logout\_button\_main\_window = utility.get\_button(self.main\_window, 'Sair', 'red', self.logout)
* self.logout\_button\_main\_window.place(x=1150, y= 550)
* self.logout\_button\_main\_window.config(cursor="hand2")
* self.view\_user\_button\_main\_window = utility.get\_button(self.main\_window, 'Visualizar usuários', 'gray',
* self.view\_user, fg = 'black')
* self.view\_user\_button\_main\_window.place(x=1150, y=450)
* self.view\_user\_button\_main\_window.config(cursor="hand2")
* self.register\_new\_user\_button\_main\_window = utility.get\_button(self.main\_window, 'Registrar novo usuário', 'gray',
* self.register\_new\_user, fg = 'black')
* self.register\_new\_user\_button\_main\_window.place(x=1150, y=350)
* self.register\_new\_user\_button\_main\_window.config(cursor="hand2")
* self.db\_dir = './db'
* if not os.path.exists(self.db\_dir):
* os.mkdir(self.db\_dir)
* self.log\_path = './log.txt'
* def add\_webcam(self, label):
* if 'cap' not in self.\_\_dict\_\_:
* self.cap = cv2.VideoCapture(0)
* self.\_label = label
* self.process\_webcam()
* def process\_webcam(self):
* ret, frame = self.cap.read()
* self.most\_recent\_capture\_arr = frame
* img\_ = cv2.cvtColor(self.most\_recent\_capture\_arr, cv2.COLOR\_BGR2RGB)
* self.most\_recent\_capture\_pil = Image.fromarray(img\_)
* imgtk = ImageTk.PhotoImage(image=self.most\_recent\_capture\_pil)
* self.\_label.imgtk = imgtk
* self.\_label.configure(image=imgtk)
* self.\_label.after(20, self.process\_webcam)
* def logout(self):
* exit()
* def register\_new\_user(self):
* self.register\_new\_user\_window = tk.Toplevel(self.main\_window)
* self.register\_new\_user\_window.geometry("1200x620+350+100")
* self.register\_new\_user\_window.overrideredirect(False)
* self.text\_label\_register\_new\_user\_window\_title = tk.Label(self.register\_new\_user\_window, text = 'Registro de usuários', font=("Arial", 30, "bold"))
* self.text\_label\_register\_new\_user\_window\_title.place(x = 400, y = 40)
* self.accept\_button\_register\_new\_user\_window = utility.get\_button(self.register\_new\_user\_window, 'Aceitar', 'green', self.accept\_register\_new\_user)
* self.accept\_button\_register\_new\_user\_window.place(x=750, y=400)
* self.accept\_button\_register\_new\_user\_window.config(cursor="hand2")
* self.try\_again\_button\_register\_new\_user\_window = utility.get\_button(self.register\_new\_user\_window, 'Tentar Novamente', 'red', self.try\_again\_register\_new\_user)
* self.try\_again\_button\_register\_new\_user\_window.place(x=750, y=500)
* self.try\_again\_button\_register\_new\_user\_window.config(cursor="hand2")
* self.capture\_label = utility.get\_img\_label(self.register\_new\_user\_window)
* self.capture\_label.place(x=10, y=100, width=700, height=500)
* self.add\_img\_to\_label(self.capture\_label)
* self.entry\_text\_register\_new\_user = utility.get\_entry\_text(self.register\_new\_user\_window)
* self.entry\_text\_register\_new\_user.place(x = 750, y = 200)
* self.text\_label\_register\_new\_user = utility.get\_text\_label(self.register\_new\_user\_window, 'Por favor, insira o nome:')
* self.text\_label\_register\_new\_user.place(x = 750, y = 150)
* def try\_again\_register\_new\_user(self):
* self.register\_new\_user\_window.destroy()
* def add\_img\_to\_label(self, label):
* imgtk = ImageTk.PhotoImage(image = self.most\_recent\_capture\_pil)
* label.imgtk = imgtk
* label.configure(image = imgtk)
* self.register\_new\_user\_capture = self.most\_recent\_capture\_arr.copy()
* def start(self):
* self.main\_window.mainloop()
* def accept\_register\_new\_user(self):
* name = self.entry\_text\_register\_new\_user.get(1.0, "end-1c")
* embeddings = face\_recognition.face\_encodings(self.register\_new\_user\_capture)[0]
* file = open(os.path.join(self.db\_dir, '{}.pickle'.format(name)), 'wb')
* pickle.dump(embeddings, file)
* utility.msg\_box('Sucesso!', 'Usuário foi registrado com sucesso !')
* self.register\_new\_user\_window.destroy()
* def view\_user(self):
* self.view\_user\_window = tk.Toplevel(self.main\_window)
* self.view\_user\_window.geometry("1200x620+350+100")
* self.view\_user\_window.overrideredirect(False)
* self.view\_user\_window\_title = tk.Label(self.view\_user\_window, text = 'Visualização dos usuários', font=("Arial", 30, "bold"))
* self.view\_user\_window\_title.place(x = 370, y = 40)
* self.view\_user\_window.grab\_set()
* self.view\_user\_window.attributes("-topmost", True)
* self.delete\_button\_view\_user\_window = utility.get\_small\_button(self.view\_user\_window, 'Deletar usuário', 'gray', self.delete\_user, fg = 'black')
* self.delete\_button\_view\_user\_window.place(x=425, y=485)
* self.delete\_button\_view\_user\_window.config(cursor="hand2")
* self.return\_button\_view\_user\_window = utility.get\_small\_button(self.view\_user\_window, 'Voltar', 'red', self.return\_edit\_user)
* self.return\_button\_view\_user\_window.place(x=425, y=550)
* self.return\_button\_view\_user\_window.config(cursor="hand2")
* flist = os.listdir(db\_dir)
* self.lbox = tk.Listbox(self.view\_user\_window, height = 15, width = 35)
* self.lbox.pack(padx=10, pady=100)
* bolded = font.Font(weight='bold', size=16)
* self.lbox.config(font=bolded, cursor = "hand2", background="lightgray")
* for item in flist:
* self.lbox.insert(tk.END, item)
* def return\_edit\_user(self):
* self.view\_user\_window.destroy()
* def delete\_user(self):
* selected\_index = self.lbox.curselection()
* if selected\_index:
* name = self.lbox.get(selected\_index)
* index = int(selected\_index[0])
* path = 'db/'
* file = path + name
* self.view\_user\_window.attributes('-disabled', 1)
* escolha = messagebox.askyesno(parent = self.view\_user\_window,title = 'Confirmação', message = 'Deseja realmente excluir esse usuário?')
* self.view\_user\_window.attributes('-disabled', 0)
* self.view\_user\_window.focus\_force()
* if escolha:
* self.lbox.delete(index)
* os.remove(file)
* if \_\_name\_\_ == "\_\_main\_\_":
* app = App()
* app.start()
* Código “Login.py”, responsável por toda a operação de reconhecimento facial e controle do acesso, bem como comunicação com a plataforma Arduíno.
* import os.path
* import tkinter as tk
* import cv2
* from PIL import Image, ImageTk
* import utility
* from time import sleep
* from pyfirmata import Arduino, util
* import time
* servo\_pin = 10
* LedLiberado\_pin = 5
* reed\_pin = 8
* PORT = 'COM3'
* board = Arduino(PORT)
* it = util.Iterator(board)
* it.start()
* servo = board.get\_pin('d:{}:s'.format(10))
* led1 = board.digital[5]
* db\_dir = './db'
* class App:
* def \_\_init\_\_(self):
* self.main\_window = tk.Tk()
* self.main\_window.geometry("1200x520+350+100")
* self.webcam\_label = utility.get\_img\_label(self.main\_window)
* self.webcam\_label.place(x=10, y=0, width=700, height=500)
* self.add\_webcam(self.webcam\_label)
* self.sensor()
* self.db\_dir = './db'
* if not os.path.exists(self.db\_dir):
* os.mkdir(self.db\_dir)
* self.log\_path = './log.txt'
* def add\_webcam(self, label):
* if 'cap' not in self.\_\_dict\_\_:
* self.cap = cv2.VideoCapture(0)
* self.\_label = label
* self.process\_webcam()
* def process\_webcam(self):
* ret, frame = self.cap.read()
* self.most\_recent\_capture\_arr = frame
* img\_ = cv2.cvtColor(self.most\_recent\_capture\_arr, cv2.COLOR\_BGR2RGB)
* def login(self):
* temp\_img = './temp.jpg'
* self.process\_webcam()
* cv2.imwrite(temp\_img, self.most\_recent\_capture\_arr)
* name = utility.recognize(self.most\_recent\_capture\_arr, db\_dir)
* while name in ['unknown\_person', 'no\_persons\_found']:
* print('Ops...', 'Usuário desconhecido. Por favor, registre-se ou tente novamente.')
* self.process\_webcam()
* temp\_img = './temp.jpg'
* cv2.imwrite(temp\_img, self.most\_recent\_capture\_arr)
* name = utility.recognize(self.most\_recent\_capture\_arr, db\_dir)
* time.sleep(5)
* else:
* servo.write(0)
* led1.write(1)
* print('Acesso permitido !', 'Bem-vindo, {}.'.format(name))
* os.remove(temp\_img)
* time.sleep(10)
* def sensor(self):
* reed\_pin\_mode = board.get\_pin('d:{}:i'.format(reed\_pin))
* reed\_pin\_mode.enable\_reporting()
* while True:
* # Leia o estado do pino do reed switch (0 para aberto, 1 para fechado)
* self.reed\_state = reed\_pin\_mode.read()
* if self.reed\_state == 1:
* print("Porta fechada")
* led1.write(0)
* servo.write(180)
* self.login()
* else:
* print("Porta aberta")
* led1.write(1)
* time.sleep(0.1)  # Pequeno atraso para evitar leituras rápidas repetidas
* if \_\_name\_\_ == "\_\_main\_\_":
* app = App()
* app.start()

* Código “Requirements.txt”, as principais bibliotecas utilizadas no processo de reconhecimento que precisarão ser instaladas pelo usuário.

cmake==3.17.2

dlib==19.18.0

opencv-python==4.6.0.66

Pillow==9.2.0

face\_recognition==1.3.0

Todos os códigos fonte, bem como instruções mais detalhadas e imagens do projeto, se encontram no seguinte repositório no github:

<https://github.com/BernardoHalfeld/ProjetoLIA>

As informações utilizadas no projeto para a implementação da biblioteca face\_recognition, foram retiradas no seguinte repositório:

<https://github.com/ageitgey/face_recognition>