

Monitoramento Automatizado de Indivíduos em Situação de Vulnerabilidade

Visão Computacional
Prof. Dr. Celso S. Kurashima

Os observadores:

Jorge Luiz Pinto Junior - RA: 11058715 - CEO

Marcos Baldrigue Andrade - RA: 11201921777 - CFO - Financeiro

Guilherme Eduardo Pereira - RA: 11201720498 - CPO - Desenvolvimento

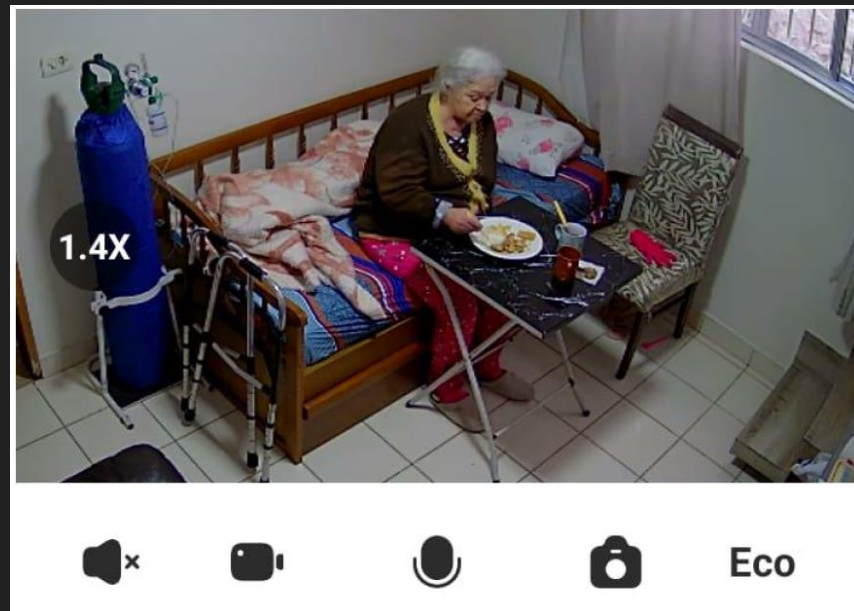
Introdução

◆ A motivação para o projeto surgiu a partir de uma entrevista com a mãe de um dos alunos.

◆ Ela relatou um caso real: sua mãe, diagnosticada com Alzheimer e demência, se levantou da cama e foi ao banheiro sozinha enquanto ela monitorava por câmera, sem receber nenhum alerta.

◆ O principal desafio relatado foi a ausência de notificações automáticas durante movimentos importantes, especialmente à noite.

importantes, especialmente à noite.



Objetivo geral do programa

- ◆ O projeto visa desenvolver um sistema que detecta automaticamente, através da biblioteca Mediapipe, quando um indivíduo (ex: idoso) se levanta da cama, sofá ou cadeira.
- ◆ Ao detectar o movimento, o sistema envia um alerta imediato ao cuidador, via aplicativo, som ou mensagem.
- ◆ A proposta contribui para aumentar a segurança em residências e instituições, especialmente quando o cuidador não está por perto.



O que o programa faz de fato:

O programa captura vídeo em tempo real, estima a pose humana com o MediaPipe, classifica a postura da pessoa em “Em pé”, “Sentada” ou “Deitada”, corrige a distorção da câmera a partir de parâmetros de calibração intrínseca e, com base em janelas temporais, envia alertas pelo Telegram quando:

- (i) alguém permanece em pé por ≥ 5 s ou
- (ii) nenhuma pessoa é detectada por ≥ 5 s.

O resultado é exibido em uma janela com a pose anotada e um rótulo textual.

Apresentação do script

1. Dependências, configuração e variáveis de ambiente

```
import cv2, time, mediapipe as mp, numpy as np, math, requests, os
from dotenv import load_dotenv

load_dotenv()
TOKEN = os.getenv('TELEGRAM_TOKEN')
CHAT_ID = os.getenv('TELEGRAM_CHAT_ID')
MENSAGEM = 'Pessoa observada esta de pé ou saiu do alcance de visão'
```

Apresentação do script

2. Calibração da câmera e correção da distorção

```
fs = cv2.FileStorage("calibration.xml", cv2.FILE_STORAGE_READ)
camera_matrix = fs.getNode("camera_matrix").mat()
dist_coeffs = fs.getNode("distortion_coefficients").mat()
fs.release()
```

Apresentação do script

3. Rotina de envio de mensagens ao telegram

```
def enviar_mensagem(texto):  
    url = f'https://api.telegram.org/bot{TOKEN}/sendMessage'  
    payload = {'chat_id': CHAT_ID, 'text': texto}  
    response = requests.post(url, data=payload)  
    ...
```


Apresentação do script

4. Geometria: cálculo de ângulo articular

```
def calculate_angle(a, b, c):  
    a, b, c = np.array(a), np.array(b), np.array(c)  
    ba, bc = a - b, c - b  
    ...  
    cos_ang = np.dot(ba, bc) / (||ba|| * ||bc||)  
    return degrees(arccos(clamp(cos_ang, -1, 1)))
```

$$\cos \theta = \frac{\vec{BA} \cdot \vec{BC}}{\|\vec{BA}\| \|\vec{BC}\|}$$

Apresentação do script

5. Classificador de postura baseado em landmarks

```
def classify_pose(landmarks):  
    xs = [lm.x ...]; ys = [lm.y ...]  
    width, height = max(xs)-min(xs), max(ys)-min(ys)  
    if width > height * 1.3:  
        return 'Deitada'  
    ...  
    left_angle = calculate_angle(left_hip, left_knee, left_ankle)  
    right_angle = calculate_angle(right_hip, right_knee, right_ankle)  
    if (left_angle + right_angle)/2.0 < 160:  
        return 'Sentada'  
    return 'Em pé'
```

Apresentação do script

6. Inicialização do estimador de pose e do desenho

```
mp_pose = mp.solutions.pose  
pose = mp_pose.Pose(static_image_mode=False, model_complexity=1, enable_segmentation=False)  
mp_drawing = mp.solutions.drawing_utils
```

Apresentação do script

7. Captura de vídeo e verificação de dispositivo

```
cap = cv2.VideoCapture(0)
if not cap.isOpened():
    raise RuntimeError('Não foi possível acessar a webcam.')
```

Apresentação do script

8. Lógica de monitoramento temporal e laço principal

```
em_pe_start = None      # instante (epoch) do início do estado "Em pé"  
ausente_start = None    # instante do início do estado "Ausente" (sem pose)  
em_pe_alertado = False  # se já alertou "Em pé  $\geq$  5 s"  
ausente_alertado = False # se já alertou "Ausente  $\geq$  5 s"
```

Apresentação do script

8. Lógica de monitoramento temporal e laço principal - Aquisição e pré-processamento por frame

```
ret, frame = cap.read()
frame_undistorted = cv2.undistort(frame, camera_matrix, dist_coeffs)
results = pose.process(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
annotated = frame.copy()
label = 'Ausente'
```

Apresentação do script

8. Lógica de monitoramento temporal e laço principal - Caso com pose detectada

```
if results.pose_landmarks:
    mp_drawing.draw_landmarks(...)
    label = classify_pose(results.pose_landmarks.landmark)
    ausente_start = None; ausente_alertado = False
    if label == 'Em pé':
        if em_pe_start is None:
            em_pe_start = time.time()
        elif not em_pe_alertado and (time.time() - em_pe_start) >= 5:
            enviar_mensagem("Pessoa está em pé por 5 segundos!")
            em_pe_alertado = True
    else:
        em_pe_start = None; em_pe_alertado = False
```

Apresentação do script

8. Lógica de monitoramento temporal e laço principal - Caso sem pose detectada (ausência)

```
else:
    em_pe_start = None; em_pe_alertado = False
    if ausente_start is None:
        ausente_start = time.time()
    elif not ausente_alertado and (time.time() - ausente_start) >= 5:
        enviar_mensagem("Ninguém detectado na câmera por 5 segundos!")
        ausente_alertado = True
```


Apresentação do script

8. Lógica de monitoramento temporal e laço principal - Sobreposição de rótulo e saída interativa

```
cv2.putText(annotated, label, (20,30), ...)  
cv2.imshow('Pose Detection (press q to quit)', annotated)  
if cv2.waitKey(1) & 0xFF == ord('q'):  
    break
```

Apresentação do script

8. Lógica de monitoramento temporal e laço principal - Liberação de recursos

```
cap.release(); cv2.destroyAllWindows(); pose.close()
```

Apresentação do script

9. Resumo conceitual por etapas

- **Carregamento de segredos (dotenv)** → garante segurança operacional (TOKEN/CHAT_ID fora do código).
- **Leitura de calibração (OpenCV)** → fornece parâmetros intrínsecos para undistort do vídeo.
- **Inicialização do MediaPipe Pose** → habilita estimativa esquelética confiável em streaming.
- **Laço de captura** → obtém frames e (idealmente) corrige distorção.
- **Estimativa de landmarks** → extrai 33 pontos anatômicos por pessoa.
- **Classificação de postura** → aplica heurísticas geométricas (razão largura/altura; ângulos de joelho).
- **Gestão temporal de eventos** → usa cronômetros para detectar permanências de 5 s em estados de interesse.
- **Notificação remota (Telegram)** → envia alertas sob condições satisfeitas.
- **Renderização** → desenha esqueleto e rótulo para feedback do operador.
- **Finalização** → libera recursos de câmera, janelas e modelo.

Aplicação prática

Durante o seminário de 13/08 foram feitos 14 testes do projeto com diferentes pessoas da turma, após a apresentação as pessoas responderam o seguinte formulário:

Aplicação prática

Durante o seminário de 13/08 foram feitos 14 testes do projeto com diferentes pessoas da turma, após a apresentação as pessoas responderam o seguinte formulário:

1. Você participou do teste do projeto? Respostas: Sim (100%)

Aplicação prática

Durante o seminário de 13/08 foram feitos 14 testes do projeto com diferentes pessoas da turma, após a apresentação as pessoas responderam o seguinte formulário:

1. Você participou do teste do projeto? Respostas: Sim (100%)
2. Você calibrou a câmera? Respostas: Sim (50%)

Aplicação prática

Durante o seminário de 13/08 foram feitos 14 testes do projeto com diferentes pessoas da turma, após a apresentação as pessoas responderam o seguinte formulário:

1. Você participou do teste do projeto? Respostas: Sim (100%)
2. Você calibrou a câmera? Respostas: Sim (50%)
3. O projeto conseguiu identificar o seu corpo? Respostas: Sim (100%)

Aplicação prática

Durante o seminário de 13/08 foram feitos 14 testes do projeto com diferentes pessoas da turma, após a apresentação as pessoas responderam o seguinte formulário:

1. Você participou do teste do projeto? Respostas: Sim (100%)
2. Você calibrou a câmera? Respostas: Sim (50%)
3. O projeto conseguiu identificar o seu corpo? Respostas: Sim (100%)
4. O projeto sinalizou quando você estava de pé por mais de 5 segundos? Sim (100%)

Aplicação prática

Durante o seminário de 13/08 foram feitos 14 testes do projeto com diferentes pessoas da turma, após a apresentação as pessoas responderam o seguinte formulário:

1. Você participou do teste do projeto? Respostas: Sim (100%)
2. Você calibrou a câmera? Respostas: Sim (50%)
3. O projeto conseguiu identificar o seu corpo? Respostas: Sim (100%)
4. O projeto sinalizou quando você estava de pé por mais de 5 segundos? Sim (100%)
5. O projeto sinalizou quando você estava ausente por mais de 5 segundos? Sim (50%)

Aplicação prática

Durante o seminário de 13/08 foram feitos 14 testes do projeto com diferentes pessoas da turma, após a apresentação as pessoas responderam o seguinte formulário:

1. Você participou do teste do projeto? Respostas: Sim (100%)
2. Você calibrou a câmera? Respostas: Sim (50%)
3. O projeto conseguiu identificar o seu corpo? Respostas: Sim (100%)
4. O projeto sinalizou quando você estava de pé por mais de 5 segundos? Sim (100%)
5. O projeto sinalizou quando você estava ausente por mais de 5 segundos? Sim (50%)
6. De 0 (não recomendaria) a 10 (recomendaria) quanto você recomendaria esse projeto para um amigo? Resposta média: 9.93 (majoritariamente aceito)

Aplicação prática

Durante o seminário de 13/08 foram feitos 14 testes do projeto com diferentes pessoas da turma, após a apresentação as pessoas responderam o seguinte formulário:

7. Sugestões:

Aplicação prática



Durante o seminário de 13/08 foram feitos 14 testes do projeto com diferentes pessoas da turma, após a apresentação as pessoas responderam o seguinte formulário:

7. Sugestões:

Não

Sugestão: cadastrar mais possíveis

Projeto excelente, nada a declarar sobre melhorias.

Top demais

Trabalhar para mais de uma pessoa.

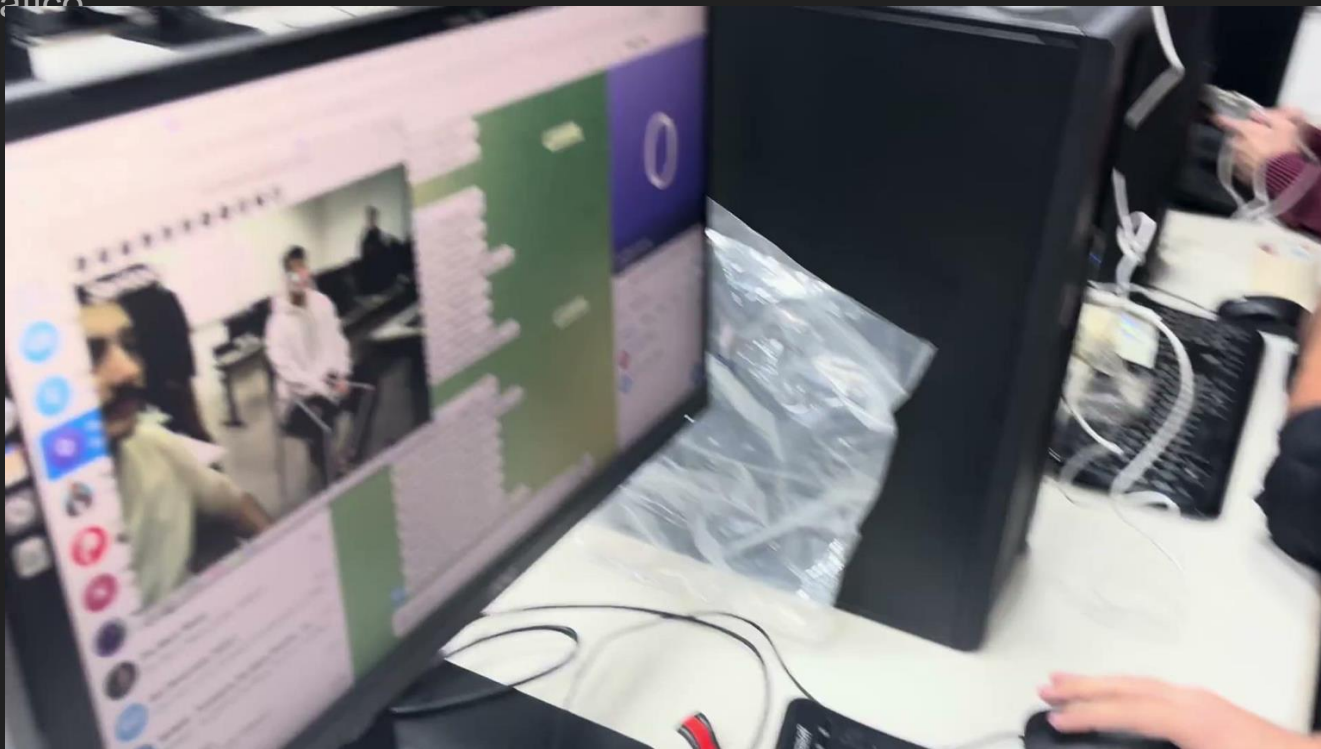
Comercializa

Não, o projeto está completo.

Nenhuma, ótimo projeto.

Aplicação prática

Vídeo prático



Conclusões finais

Requisitos solicitados:

| | |
|------------------------|--|
| i. Obrigatórios | <ul style="list-style-type: none">• Filtragem de imagens• Transformações geométricas• Calibração de câmeras• Propriedades Intrinsecas e extrinsecas |
| ii. Pelo menos um item | <ul style="list-style-type: none">• Estereoscopia• Profundidade• Reconhecimento• Rastreamento |

Conclusões finais

Requisitos solicitados:

1. Filtragem de imagens

a. Como foi atendido

- i. O código aplica uma operação de filtragem indireta por meio da correção de distorção da câmera:

```
frame_undistorted = cv2.undistort(frame, camera_matrix, dist_coeffs)
```
- ii. Essa operação é um tipo de filtragem geométrica aplicada sobre a imagem, removendo ruídos causados pela lente (efeito olho de peixe, distorções radiais/tangenciais).

Conclusões finais

Requisitos solicitados:

2. Transformações geométricas

a. Como foi atendido

- i. A correção de distorção feita por: `cv2.undistort(frame, camera_matrix, dist_coeffs)`
- ii. É uma transformação geométrica, pois mapeia cada ponto da imagem original para uma nova posição corrigida.
- iii. Além disso, o cálculo de ângulos entre pontos do corpo também se baseia em transformações geométricas (vetores, produto escalar, ângulo em segmentos).

Conclusões finais

Requisitos solicitados:

3. Calibração de câmeras

a. Como foi atendido

- i. O programa lê os parâmetros de calibração a partir do arquivo calibration.xml:
- ii. Esses parâmetros vêm de uma calibração prévia (padrão xadrez OpenCV)

```
camera_matrix = fs.getNode("camera_matrix").mat()  
dist_coeffs = fs.getNode("distortion_coefficients").mat()
```

Conclusões finais

Requisitos solicitados:

4. Propriedade Intrínsecas e Extrínsecas

a. Como foi atendido

- i. Intrínsecas: estão no `camera_matrix` (focais f_x , f_y , centro ótico c_x , c_y) e nos `dist_coeffs` (distorções).
- ii. Extrínsecas: o `mediaPipe` internamente gera `landmarks` normalizados que dependem implicitamente da posição relativa da câmera (pose estimation).

Conclusões finais

Requisitos solicitados:

5. Reconhecimento

a. Como foi atendido

i. O código implementa reconhecimento de postura:

```
label = classify_pose(results.pose_landmarks.landmark)
```

Conclusões finais

Requisitos solicitados:





6. Rastreamento

a. Atendido parcialmente





- i. O MediaPipe Pose em **static_image_mode=False** faz rastreamento temporal dos landmarks entre quadros.
- ii. Além disso, o código implementa um rastreamento de estado (cronômetro de "em pé" e "ausente") para detectar condições persistentes. Ou seja, há rastreamento tanto de landmarks no tempo quanto de estado da pessoa.

Conclusões finais - resumo

Obrigatórios:

- Filtragem de imagens →  (correção de distorção)
- Transformações geométricas →  (undistort + cálculo de ângulos)
- Calibração de câmeras →  (uso de camera_matrix e dist_coeffs)
- Propriedades intrínsecas/extrínsecas →  (intrínsecas claras; extrínsecas indiretas)

Opcionais (pelo menos um):

- Estereoscopia → 
- Profundidade → 
- Reconhecimento →  (classificação de postura)
- Rastreamento →  (pose tracking + controle temporal de estados)



Portanto, o código atende plenamente os requisitos obrigatórios e também atende os opcionais via Reconhecimento e Rastreamento.

Referências

- LEARNOPENCV. Building a Body Posture Analysis System using MediaPipe. Disponível em: <https://learnopencv.com/building-a-body-posture-analysis-system-using-mediapipe/>. Acesso em: 7 jul. 2025.
- LEARNOPENCV. Geometry of Image Formation. Disponível em: <https://learnopencv.com/geometry-of-image-formation/>. Acesso em: 2 jul. 2025.
- LEARNOPENCV. Camera Calibration using OpenCV. Disponível em: <https://learnopencv.com/camera-calibration-using-opencv/>. Acesso em: 2 jul. 2025.