

# CIÊNCIAS DA COMPUTAÇÃO

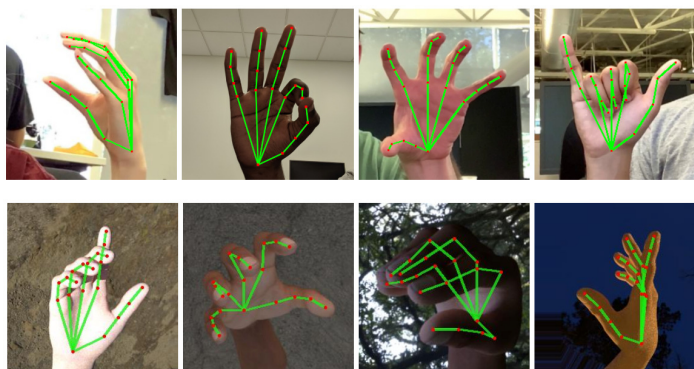
## Processamento de Imagens e Visão Computacional

Prof. César C. Xavier

1

## Rastreamento de Mãos

- Principais etapas de um sistema rastreamento mãos
- Detecção Mão e Criação dos Pontos de Referência
- Exemplo Prático



Prof. César C. Xavier

2

## ROTEIRO

- Rastreamento da Mão
  - Media Pipe – Google
  - Carregando Informações Mãos
  - Marcas de Referência e QPS
  - Extração Informações Marcas de Referência
  - Identificando as Marcas de Referência
  - Criando Módulo Extração das Referências
- Mais Exemplos
  - Contador de Dedos
  - Desenhando com os Dedos




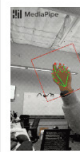



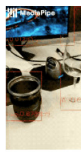




Prof. César C. Xavier

3

## Media Pipe - Google

ML solutions in MediaPipe

O MediaPipe oferece soluções de ML personalizáveis e multiplataforma para mídia ao vivo e streaming.

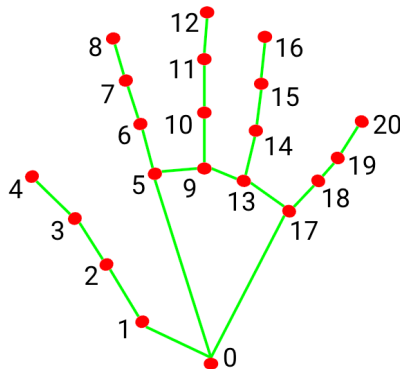
Face Detection	Face Mesh	Iris	Hands	Pose	Holistic
					
Hair Segmentation	Object Detection	Box Tracking	Instant Motion Tracking	Objectron	KNIFT
					
	Android	iOS	C++	Python	JS
Face Detection	✓	✓	✓	✓	✓
Face Mesh	✓	✓	✓	✓	✓

Prof. César C. Xavier

4

## Media Pipe - Google

- Pontos de Referência (Landmarks)



0. WRIST	11. MIDDLE_FINGER_DIP
1. THUMB_CMC	12. MIDDLE_FINGER_TIP
2. THUMB_MCP	13. RING_FINGER_MCP
3. THUMB_IP	14. RING_FINGER_PIP
4. THUMB_TIP	15. RING_FINGER_DIP
5. INDEX_FINGER_MCP	16. RING_FINGER_TIP
6. INDEX_FINGER_PIP	17. PINKY_MCP
7. INDEX_FINGER_DIP	18. PINKY_PIP
8. INDEX_FINGER_TIP	19. PINKY_DIP
9. MIDDLE_FINGER_MCP	20. PINKY_TIP
10. MIDDLE_FINGER_PIP	

<https://google.github.io/mediapipe/solutions/hands>

## Carregando Informações Mãos

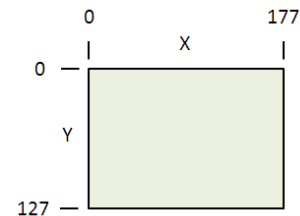
- Landmarks
  - são pontos-chave ou posições específicas em uma imagem ou objeto que são identificados e rastreados em tarefas de visão computacional.
  - referem-se a pontos específicos e significativos nas mãos, como as articulações e extremidades dos dedos, que são usados para definir a estrutura da mão.
  - da mão são 21 pontos de referência distribuídos em partes importantes da mão, como as articulações dos dedos e a palma.
  - são definidos em um modelo tridimensional (com coordenadas X, Y e Z) e fornecem uma representação detalhada da posição e orientação da mão na imagem ou vídeo.

## Carregando Informações Mãos

- Landmarks

- Coordenadas:

- X: Representa a posição horizontal do ponto na imagem (normalizada entre 0.0 e 1.0).
    - Y: Representa a posição vertical do ponto na imagem (também normalizada entre 0.0 e 1.0).
    - Z: Representa a profundidade (ou quão próximo o ponto está da câmera). Não é normalizado e é relativo à escala da imagem.



Prof. César C. Xavier

## Carregando Informações Mãos

- Landmarks

- Aplicações:

- Rastreamento de Gestos
      - Com a posição dos landmarks, é possível rastrear movimentos e realizar a detecção de gestos com alta precisão, como identificar quando os dedos estão dobrados ou estendidos.
    - Animação e Realidade Aumentada
      - Os landmarks permitem a criação de modelos de mãos animados ou interações em aplicações de realidade aumentada.
    - Interpretação de Gestos
      - Usando a posição dos landmarks, é possível interpretar gestos específicos, como "pinça", "apontar" ou "fechar o punho".

Prof. César C. Xavier

## Carregando Informações Mãos

- Objeto **mpHands** = mp.solutions.hands
  - é uma referência ao módulo mediapipe.solutions.hands
  - utilizada para detecção e rastreamento de mãos
  - contém todas as classes, funções e métodos relacionados ao processo de detecção de mãos oferecidos pela Mediapipe

## Carregando Informações Mãos

- Objeto **hands** = mpHands.Hands()
  - é uma instância da classe Hands presente no módulo mp.solutions.hands
  - é criado um objeto que executa efetivamente a detecção de mãos em imagens ou vídeos.

## Carregando Informações Mãos

- Objeto **mpDraw** = `mp.solutions.drawing_utils`
  - contém funções utilitárias que facilitam o desenho de landmarks (pontos) e conexões em imagens ou vídeos, como os marcos de mãos, rosto ou corpo detectados pelo Mediapipe.
  - função do `drawing_utils` é desenhar os pontos e as conexões que o Mediapipe detecta nas imagens
  - permite que os resultados da detecção sejam visualizados de maneira mais clara em vídeos ou imagens.

## Carregando Informações Mãos

- `results = hands.process(imgRGB)`
  - `hands`:
    - é uma instância da classe `Hands()` do módulo Mediapipe Hands.
    - é responsável por processar as imagens e detectar mãos
  - O método `process()`:
    - é uma função do objeto `hands`
    - realiza a detecção de mãos
    - Entrada:
      - `ImgRGB`:
        - uma imagem que está sendo processada, já convertida para o espaço de cores RGB (normalmente a captura de uma webcam está em BGR, então é necessário converter para RGB, o que deve ser feito antes desta linha).
        - essa imagem é usada como entrada para o modelo de detecção de mãos.
    - Saída:
      - `results`:
        - o método `process()` retorna um objeto contendo os resultados da detecção de mãos.
        - esse objeto contém várias informações, como os landmarks (pontos de referência das mãos) e a quantidade de mãos detectadas.

## Carregando Informações Mãos

```
import cv2
import mediapipe as mp
import time

cap = cv2.VideoCapture(0)

#
# Importa API
#
mpMaos = mp.solutions.hands

# Criando objeto mao

# class Hands(static_image_mode=False,
#             max_num_hands=2,
#             model_complexity=1,
#             min_detection_confidence=0.5,
#             min_tracking_confidence=0.5)
# static_image_mode: Se a entrada for uma imagem única, definimos
# como true, caso contrário, configuramos false para rastrear quadros
# max_num_hands: número máximo de mãos no quadro, padrão 2
# model_complexity: Dois modelos 0 ou 1 onde 1 fornece melhores
# resultados que 0
# min_detection_confidence: confiança das detecções
# min_tracking_confidence: se rastreando quadros, rastreando a
# confiança

maos = mpMaos.Hands()

while True:
    success, img = cap.read()
    imgRGB = cv2.cvtColor(img,
cv2.COLOR_BGR2RGB)
    resultado = maos.process(imgRGB)

    if resultado.multi_hand_landmarks:
        print(resultado.multi_hand_landmarks)

    cv2.imshow("Camera", img)

    key = cv2.waitKey(1)
    if key == ord('q'):
        break
    if
cv2.getWindowProperty('Camera',cv2.WND_PROP
_VISIBLE) <= 0:
        break
    cv2.destroyAllWindows()
```

Prof. César C. Xavier

13

## Desenhando Marcas de Referência e #Quadros por Segundo

```
.
aTempo = 0 # tempo anterior
pTempo = 0 # tempo presente

#
# Método para desenhar as referencias
#

mpDraw = mp.solutions.drawing_utils

while True:
    success, img = cap.read()
    imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    resultado = maos.process(imgRGB)

    if resultado.multi_hand_landmarks:
        #print(resultado.multi_hand_landmarks)
        for maosLista in resultado.multi_hand_landmarks:
            mpDraw.draw_landmarks(img, maosLista, mpMaos.HAND_CONNECTIONS)

    pTempo = time.time()
    fps = 1/(pTempo - aTempo)
    aTempo = pTempo
    cv2.putText(img, str(int(fps)), (10,40), cv2.FONT_HERSHEY_PLAIN, 2, (255,0,255), 3 )

    cv2.imshow("Camera", img)

    .
    .
    .
```

Prof. César C. Xavier

14

## Extração Informações Marcas de Referência

```
.  
. .  
while True:  
    success, img = cap.read()  
    imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  
    resultado = maos.process(imgRGB)  
  
    if resultado.multi_hand_landmarks:  
        for maosLista in resultado.multi_hand_landmarks:  
            for id, mr in enumerate(maosLista.landmark):  
                print(id, mr)  
                mpDraw.draw_landmarks(img, maosLista, mpMaos.HAND_CONNECTIONS)  
  
    pTempo = time.time()  
    fps = 1/(pTempo - aTempo)  
    aTempo = pTempo  
    cv2.putText(img, str(int(fps)), (10,40), cv2.FONT_HERSHEY_PLAIN, 2, (255,0,255), 3 )  
. . .
```

## Extração Informações Marcas de Referência

Coordenadas Tela:

```
.  
. .  
while True:  
    success, img = cap.read()  
    imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  
    resultado = maos.process(imgRGB)  
  
    if resultado.multi_hand_landmarks:  
        for maosLista in resultado.multi_hand_landmarks:  
            for id, mr in enumerate(maosLista.landmark):  
                h, w, c = img.shape  
                cx, cy = int(mr.x*w), int(mr.y*h)  
                print(id, cx, cy)  
                mpDraw.draw_landmarks(img, maosLista, mpMaos.HAND_CONNECTIONS)  
  
    pTempo = time.time()  
    fps = 1/(pTempo - aTempo)  
    aTempo = pTempo  
    cv2.putText(img, str(int(fps)), (10,40), cv2.FONT_HERSHEY_PLAIN, 2, (255,0,255), 3 )  
. . .
```



## Identificando Marcas de Referência

```
.
.

while True:
    success, img = cap.read()
    imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    resultado = maos.process(imgRGB)

    if resultado.multi_hand_landmarks:
        for maosLista in resultado.multi_hand_landmarks:
            for id, mr in enumerate(maosLista.landmark):
                h, w, c = img.shape
                cx, cy = int(mr.x*w), int(mr.y*h)
                print(id, cx, cy)
            mpDraw.draw_landmarks(img, maosLista, mpMaos.HAND_CONNECTIONS)

    pTempo = time.time()
    fps = 1/(pTempo - aTempo)
    aTempo = pTempo
    cv2.putText(img, str(int(fps)), (10,40), cv2.FONT_HERSHEY_PLAIN, 2, (255,0,255), 3 )
.
```

## Criação Módulo Extração das Referências

```
import cv2
import mediapipe as mp
from cv2 import cv2
import time

class deteccaoMao():
    def __init__(self, mode=False, nummaxMaos=2, complexidade=1,
    certezadeteccao=0.5, certezarastreamento=0.5):
        self.mode = mode
        self.nummaxMaos = nummaxMaos
        self.complexidade = complexidade
        self.certezadeteccao = certezadeteccao
        self.certezarastreamento = certezarastreamento

#
# Importa API
#
    self.mpMaos = mp.solutions.hands

#
# Criando objeto mao
#

# class Hands(static_image_mode=False,
# max_num_hands=2,
# model_complexity=1,
# min_detection_confidence=0.5,
# min_tracking_confidence=0.5)

    self.maos = self.mpMaos.Hands(self.mode, self.nummaxMaos,
    self.complexidade, self.certezarastreamento, self.certezadeteccao)

    self.mpDraw = mp.solutions.drawing_utils

def encontreMaos(self, img, desenhar=True):
    imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    self.resultado = self.maos.process(imgRGB)

    if self.resultado.multi_hand_landmarks:
        for maosLista in self.resultado.multi_hand_landmarks:
            if desenhar:
                self.mpDraw.draw_landmarks(img, maosLista,
                self.mpMaos.HAND_CONNECTIONS)

    return img

def encontreReferencias(self, img, numMao=0, desenhar=False):
    listaReferencias = []
    if self.resultado.multi_hand_landmarks:
        minhaMao = self.resultado.multi_hand_landmarks[numMao]
        for id, mr in enumerate(minhaMao.landmark):
            h, w, c = img.shape
            cx, cy = int(mr.x*w), int(mr.y*h)
            #print(id, cx, cy)
            listaReferencias.append([id, cx, cy])
            if desenhar:
                if id == 4:
                    cv2.circle(img, (cx, cy), 20, (255, 5, 255), cv2.FILLED)
    return listaReferencias
```

## Criação Módulo Extração das Referências

```
from cv2 import cv2
import mediapipe as mp
import time
import mrm_sld11 as mrm

aTempo = 0 # tempo anterior
pTempo = 0 # tempo presente
cap = cv2.VideoCapture(0)
detector = mrm.deteccaoMao()

while True:
    success, img = cap.read()
    img = detector.encontreMaos(img)
    listaReferencias = detector.encontreReferencias(img)
    if len(listaReferencias) != 0:
        print(listaReferencias[4])

    pTempo = time.time()
    fps = 1/(pTempo - aTempo)
    aTempo = pTempo

    cv2.putText(img, str(int(fps)), (10,40), cv2.FONT_HERSHEY_PLAIN, 2, (255,0,255), 3 )

    cv2.imshow("Camera", img)

    key = cv2.waitKey(1)
    if key == ord('q'):
        break
    if cv2.getWindowProperty('Camera',cv2.WND_PROP_VISIBLE) <= 0:
        break
cv2.destroyAllWindows()
```

Prof. César C. Xavier

19

## Mais Exemplos

- Contando Dedos



Prof. César C. Xavier

20

## Mais Exemplos

- Desenhando com as Mãos



Prof. César C. Xavier