04-landmarks_position.ipynb

Pablo Munarriz Senosiain

12 de diciembre de 2024

1. Introducción

El propósito de este documento es explicar el código del cuaderno '04-landmarks_position.ipynb'. En este cuaderno de Jupyter se definen varias funciones con las que se pretende establecer relaciones entre los 'pose landmarks' de MediaPipe y varios atributos naturales de los brazos. De esta forma, trataríamos de generar el movimiento de los brazos entre dos posturas.

2. Pose Landmarks

MediaPipe ofrece, entre otras cosas, modelos de detección de puntos de referencia sobre personas. Concrétamente, el modelo 'pose lanmarker model' detecta 33 puntos sobre los cuerpos humanos, y su objetivo es detectar posturas.

Al aplicar el modelo a una imagen, obtenemos un objeto del tipo 'mediapipe.framework.formats.landmark_pb2.NormalizedLandmarkList'. Este objeto, tal y como dice el nombre, es una lista de 'landmarks normalizados'. Cada uno de estos landmarks contiene hasta cuatro atributos: 'x', 'y', 'z' y 'visibility'. Aquí podemos ver los detalles de cada atributo:

x and y: Landmark coordinates normalized between 0.0 and 1.0 by the image width (x) and height (y).

z: The landmark depth, with the depth at the midpoint of the hips as the origin. The smaller the value, the closer the landmark is to the camera. The magnitude of z uses roughly the same scale as x.

visibility: The likelihood of the landmark being visible within the image.

En nuestro caso, almacenamos los valores de estos atributos en un array de NumPy, por lo que los objetos 'NormalizedLandmarkList' acaban siendo arrays bidimensionales, donde la primera dimensión representa cada landmark y la segunda dimensión cada atributo. Concretamente, los 'pose_landmarks' se almacenan en arrays de dimensión $(33, 3)$ ó $(33, 4)$, en función de si almacenamos la visibilidad o no. El 'dtype' que usamos es 'np.float64'.

Por otra parte, para entender la posición de los brazos nos basta únicamente con los landmarks del torso y brazos, por lo que también solemos trabajar con objetos de nombre 'half_pose_landmarks', que básicamente son los

subarrays de los 'pose_landmarks' que contienen la información de los landmarks que queremos. Concretamente, la relación es 'half_pose_landmarks = pose_landmarks[11:25]'.

3. Atributos

Para entender la posición de los brazos, en primer lugar consideramos una base para cada brazo. El primer vector de la base es el vector unitario que tiene la misma dirección que el vector entre los hombros y que, al situarlo en el hombro en cuestión, apunta hacia fuera del cuerpo. El segundo vector de la base es también unitario, ortogonal al primero, contenido en el plano del torso y en sentido de los hombros a las caderas. El tercer vector es el necesario para que las bases sean ortonormales, por lo que es el producto vectorial del primer vector con el segundo. Así, el tercer vector de la base del brazo izquierdo sale del hombro ortogonal al plano de la espalda (hacia atrás), mientras que el tercer vector de la base del brazo derecho sale del hombro ortogonal al plano del torso (hacia adelante). En la Figura 1 vemos una representación gráfica de ambas bases.

También consideraremos una base para cada mano. En este caso, el primer vector de la base es el vector unitario normal a la palma de la mano (saliendo de la palma). El tercer vector será el que apunte de la muñeca hacia el centro de la palma. Y el segundo, para hacer una base ortonormal, será el producto vectorial del tercer vector con el primero. De nuevo, el sentido del segundo vector cambia en función del lado: para la mano derecha, el segundo vector apunta hacia el lado del pulgar; para la mano izquierda, al lado del meñique.

Los atributos que consideraremos para tratar de entender la posición de cada brazo serán:

- Dirección del hombro: vector unitario que apunta del hombro al codo expresado en la base del brazo.
- Longitud del brazo (superior): distancia entre el hombro y el codo.
- Rotación del hombro: en radianes, respecto a alguna posición concreta.
- Ángulo del codo: en radianes.
- Longitud del antebrazo: distancia entre el codo y la muñeca.
- Rotación del codo: en radianes, respecto a alguna posición concreta.
- Rotación e inclinación de la muñeca: coordenadas esféricas del vector unitario que representa al antebrazo en la base de la mano.
- Vectores del meñique, índice y pulgar en la base de la mano.

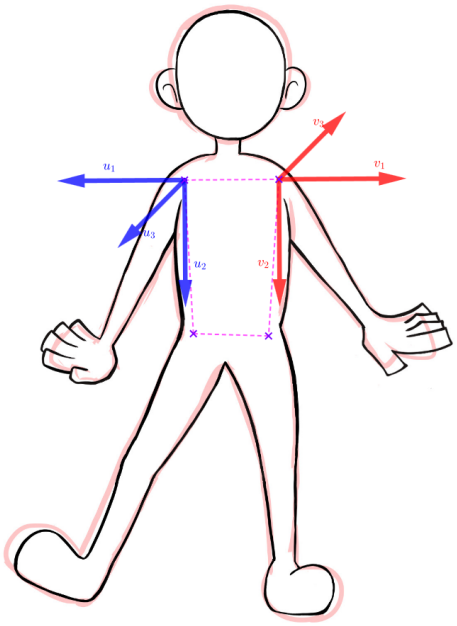

Figura 1: Representación gráfica de las bases de los brazos.

4. Preliminares

Antes de empezar con las funciones que traducen de landmarks a atributos y viceversa, vamos a comentar algunas funciones previas que conviene definir.

4.1. azimuth y colatitude

Estas funciones devuelven el ángulo acimutal y la colatitud de un vector real tridimensional y unitario, respectivamente. Tienen tres parámetros de entrada: 'x', 'y' y 'z', que representan las coordenadas del vector.

La función 'azimuth' devuelve un número en el intervalo $[-\pi, \pi)$ que mide el ángulo en radianes entre el semieje positivo X y el vector $(x, y, 0)$, siendo positivo cuando $y > 0$. Cabe destacar que cuando 'x' e 'y' son cero, la función devuelve cero.

Por otra parte, 'colatitude' devuelve un número en el intervalo $[0, \pi]$ que mide el ángulo en radianes entre el semieje positivo Z y el vector (x, y, z). En la Figura 2 tenemos el código de cada función.

4.2. max shoulder rotation

Las siguientes funciones se llaman 'max_left_shoulder_rotation' y 'max_right_shoulder_rotation'. El objetivo de estas funciones es, dada la dirección del hombro, devolver el vector normal al plano que contiene el brazo cuando el

```python
def azimuth(x, y, z):
    """
    Given a unit 3D real vector, returns the azimuth of the vector.
    """
    if np.isclose(x, 0):
        if np.isclose(y, 0):
            return np.float64(0)
        else:
            return np.sign(y) * pi / 2.
    else:
        aux = np.arctan(y/x)
        if x>0:
            return aux
        elif y>0:
            return aux + pi
        else:
            return aux - pi

def colatitude(x, y, z):
    """
    Given a unit 3D real vector, returns the colatitude of the vector.
    """
    return np.arccos(np.clip(z, -1, 1))
```

Figura 2: Código de las funciones azimuth y colatitude.

hombro está rotado al máximo (en sentido antihorario para el hombro izquierdo y horario para el hombro derecho).

Estas funciones se basan en cuatro pasos:

1. Construir un vector auxiliar que represente la dirección del antebrazo en una posición determinada.
2. Calcular el ángulo entre el vector auxiliar y la dirección del antebrazo cuando el ángulo del codo es de 90 grados y el hombro está rotado al máximo.
3. Calcular la dirección del antebrazo cuando el ángulo del codo es de 90 grados y el hombro está rotado al máximo.
4. Devolver el producto tensorial entre la dirección del hombro y la dirección del antebrazo del paso anterior.

Vayamos punto por punto, empezando por el primero. La idea para la elección del vector auxiliar es muy sencilla, simplemente tomaremos la dirección del antebrazo que es ortogonal a la del hombro (es decir, fijamos el ángulo del codo

```
if np.isclose(y, 0):
    aux_vector = np.array([0, -1, 0], dtype=np.float64)
elif np.isclose(y, 1):
    aux_vector = np.array([0, 0, -1], dtype=np.float64)
elif np.isclose(y, -1):
    aux_vector = np.array([0, 0, 1], dtype=np.float64)
else:
    aux_vector = np.sign(y) * np.array([x, y - 1./y, z], dtype=np.float64)
    aux_vector /= np.linalg.norm(aux_vector)
```

Figura 3: Código del vector auxiliar de la función 'max_left_shoulder_rotation'.

a 90 grados) y tal que las proyecciones de las dos direcciones sobre el plano $y = 0$ tienen la misma dirección. Debemos considerar algunos casos particulares: en primer lugar, si la dirección del hombro ya está sobre el plano $y = 0$, entonces el vector auxiliar será $(0, -1, 0)$ (el antebrazo apunta hacia arriba, si estamos erguidos); en segundo lugar, si la dirección del hombro es $(0, 1, 0)$ (apunta hacia el suelo, si estamos erguidos), entonces su proyección sobre el plano $y = 0$ es el vector nulo, con lo que decidimos que el vector auxiliar sea el que apunta hacia el frente, por lo que será $(0, 0, -1)$ en el brazo izquierdo y $(0, 0, 1)$ en el derecho; por último, si la dirección del hombro es $(0, -1, 0)$, decidimos que el vector auxiliar sea el que apunta hacia atrás, por lo que será $(0, 0, 1)$ en el brazo izquierdo y $(0, 0, -1)$ en el derecho.

Supongamos ahora que la dirección del hombro es (x, y, z), con y distinto de 0, 1 y -1. Si $y > 0$, entonces queremos que el vector auxiliar sea proporcional a uno de la forma (x, y', z). Como además queremos que los dos vectores sean ortogonales, debe pasar que $x^2 + yy' + z^2 = 0$, por lo que

$$y' = -\frac{x^2 + z^2}{y} = \frac{y^2 - 1}{y} = y - 1/y\,.$$

En cambio, si $y < 0$, entonces lo que queremos es que el vector auxiliar sea proporcional a uno de la forma $(-x, y', -z)$. En este caso, $-x^2 + yy' - z^2 = 0$ implica

$$y' = \frac{x^2 + z^2}{y} = \frac{1 - y^2}{y} = 1/y - y\,.$$

En definitiva, el vector auxiliar es de la forma

$$\text{signo}\,(y)\,(x, y - 1/y, z)\ .$$

En la figura 3 vemos el código que calcula este vector auxiliar en la función 'max_left_shoulder_rotation'. Cabe decir que la variable 'pi'se ha definido previamente como 'np.float64(np.pi)'.

Vamos con el segundo paso. Tenemos que obtener el ángulo entre el vector auxiliar que acabamos de definir y la dirección del antebrazo cuando el ángulo del codo es de 90 grados y el hombro está rotado al máximo. En este caso vamos

a ir brazo por brazo, empezando por el izquierdo. El ángulo lo consideraremos en sentido antihorario respecto a la dirección del hombro. La idea para obtener este ángulo, cualquiera que sea la dirección del hombro, consiste en estimar (a ojo) su valor para ciertas direcciones del hombro y asumir una especie de linealidad en distintos sectores.

En primer lugar, vamos a suponer que cuando la dirección del hombro es $(0,1,0)$, el ángulo que buscamos es $\pi/4$. Es decir, que cuando la parte superior del brazo apunta hacia el suelo (si estamos erguidos), la dirección del antebrazo cuando el ángulo del codo es de 90 grados y el hombro está rotado al máximo se separa del vector auxiliar (el que apunta hacia el frente) 45 grados.

En segundo lugar, vamos a suponer que, cuando la dirección del hombro es de la forma $(\cos\varphi, 0, \sin\varphi)$, con $\varphi \in [-\pi, 0]$, el ángulo es 0. Es decir, que cuando la dirección del hombro es paralela al suelo (si estamos erguidos) y no apunta hacia atrás, la dirección del antebrazo cuando el ángulo del codo es de 90 grados y el hombro está rotado al máximo es precisamente el vector auxiliar, por lo que apunta hacia arriba.

En tercer lugar, vamos a suponer que, cuando la dirección del hombro es $(0,0,1)$ (apunta totalmente hacia atrás), el ángulo es $-\pi/2$. Es decir, que la dirección del antebrazo cuando el ángulo del codo es de 90 grados y el hombro está rotado al máximo es $(1,0,0)$.

Por último, vamos a suponer que, cuando la dirección del hombro es $(0,-1,0)$ (apunta hacia arriba), el ángulo es 0. Es decir, que la dirección del antebrazo cuando el ángulo del codo es de 90 grados y el hombro está rotado al máximo es $(0,0,1)$ (apunta hacia atrás).

Vamos a notar por $\psi_I : S^2 \to \mathbb{R}$ a la función que, a cada dirección del hombro izquierdo, le asigna el ángulo del que estamos hablando. Las suposiciones anteriores se reducen a:

$$\begin{cases} \psi_I(0,1,0) &= \pi/4\,, \\ \psi_I(\cos\varphi, 0, \sin\varphi) &= 0\,, \text{ para cada } \varphi \in [-\pi, 0]\,, \\ \psi_I(0,0,1) &= -\pi/2\,, \\ \psi_I(0,-1,0) &= 0\,. \end{cases}$$

Lo siguiente que vamos a tratar de entender es que, si fijamos un $\varphi \in [-\pi, 0]$, entonces debe cumplirse que

$$\lim_{\theta \to 0^+} \psi_I(\sin\theta\cos\varphi, \cos\theta, \sin\theta\sin\varphi) = -(\varphi + \pi/4)\,.$$

¿Por qué? En primer lugar, vemos que estamos usando coordenadas esféricas para representar los vectores de S^2. φ es el azimut y θ es la colatitud respecto al eje Y. Entonces, al tomar el límite cuando θ tiende a 0 por la derecha, estamos yendo hacia el vector $(0,1,0)$ por el arco de azimut constante φ. Observamos que, para todos los vectores de este arco cuya colatitud es $\theta \in (0, \pi/2)$, la proyección sobre el plano $y = 0$ del vector auxiliar del paso anterior tiene la misma dirección y sentido: $(\cos\varphi, 0, \sin\varphi)$.

Así pues, al calcular el límite anterior, tiene sentido que el resultado sea el ángulo entre el vector $(\cos\varphi, 0, \sin\varphi)$ y el de la dirección del antebrazo cuando el

ángulo del codo es de 90 grados, la dirección del hombro es $(0,1,0)$ y la rotación del hombro es máxima. Teniendo en cuenta que $\psi_I(0,1,0) = \pi/4$ y el vector auxiliar de $(0,1,0)$ es $(0,0,-1)$ es fácil llegar al resultado final.

En otras palabras, el resultado del límite es el ángulo entre el límite de los vectores auxiliares y la dirección del antebrazo que debemos considerar.

Sea pues $\varphi \in [-\pi, 0]$. Consideremos la función $\psi_I^{\varphi} : [0, \pi/2] \to [-\pi, \pi)$ definida por $\psi_I^{\varphi}(\theta) = \psi_I(\sin\theta\cos\varphi, \cos\theta, \sin\theta\sin\varphi)$, para cada $\theta \neq 0$ y extendida de forma contínua a $\theta = 0$. Vamos a suponer que ψ_I^{φ} es lineal. Como $\psi_I^{\varphi}(0) = -(\varphi + \pi/4)$ y $\psi_I^{\varphi}(\pi/2) = 0$, tenemos que

$$\psi_I(\sin\theta\cos\varphi, \cos\theta, \sin\theta\sin\varphi) = \left(\varphi + \frac{\pi}{4}\right)\left(\frac{2}{\pi}\theta - 1\right), \text{ para cada } \theta \in (0, \pi/2].$$

Siguiendo la misma idea, si fijamos $\varphi \in [-\pi, 0]$, debe cumplirse que

$$\lim_{\theta \to \pi^-} \psi_I(\sin\theta\cos\varphi, \cos\theta, \sin\theta\sin\varphi) = \varphi + \pi/2.$$

Y, de nuevo, suponiendo la linealidad de la función correspondiente, obtenemos que

$$\psi_I(\sin\theta\cos\varphi, \cos\theta, \sin\theta\sin\varphi) = \left(\varphi + \frac{\pi}{2}\right)\left(\frac{2}{\pi}\theta - 1\right), \text{ para cada } \theta \in [\pi/2, \pi).$$

Hasta ahora hemos considerado las direcciones del hombro cuya tercera componente es no negativa. Ahora faltan las otras direcciones. Lo primero que debemos hacer es determinar el ángulo para todas las direcciones que son paralelas al suelo (estando erguido), es decir, aquellas para las que la segunda componente es nula. Vamos a definir la función $\psi_I' : [0, \pi) \to [-\pi, \pi)$, definida por $\psi_I'(\varphi) = \psi_I(\cos\varphi, 0, \sin\varphi)$ para cada φ. Suponemos de nuevo que esta función es lineal, así que, como $\psi_I'(0) = 0$ y $\psi_I'(\pi/2) = -\pi/2$, se tiene que $\psi_I(\cos\varphi, 0, \sin\varphi) = -\varphi$, para cada $\varphi \in [0, \pi)$.

Fijemos ahora $\varphi \in (0, \pi)$. Siguiendo la misma idea de antes, debe cumplirse que

$$\lim_{\theta \to 0^+} \psi_I(\sin\theta\cos\varphi, \cos\theta, \sin\theta\sin\varphi) = -(\varphi + \pi/4).$$

Y, de nuevo, suponiendo la linealidad de la función correspondiente, obtenemos que

$$\psi_I(\sin\theta\cos\varphi, \cos\theta, \sin\theta\sin\varphi) = -\varphi - \frac{\pi}{4} + \frac{\theta}{2}, \text{ para cada } \theta \in [\pi/2, \pi).$$

Por último, fijando de nuevo $\varphi \in (0, \pi)$, debe cumplirse que

$$\lim_{\theta \to \pi^-} \psi_I(\sin\theta\cos\varphi, \cos\theta, \sin\theta\sin\varphi) = \varphi + \pi/2.$$

Con lo que se sigue que, para cada $\theta \in [\pi/2, \pi)$,

$$\psi_I(\sin\theta\cos\varphi, \cos\theta, \sin\theta\sin\varphi) = -\varphi + \left(2\varphi + \frac{\pi}{2}\right)\left(\frac{2}{\pi}\theta - 1\right).$$

```
if np.isclose(y, 1):
    angle = pi / 4.
elif np.isclose(y, -1):
    angle = 0.
else:
    tita = colatitude(x, z, y)
    fi = azimuth(x, z, y)
    if z<=0:
        if y>=0:
            angle = (fi + pi / 4.) * (2. * tita / pi - 1.)
        else:
            angle = (fi + pi / 2.) * (2. * tita / pi - 1.)
    else:
        if y>=0:
            angle = tita / 2. - fi - pi / 4.
        else:
            angle = - fi + (2. * fi + pi / 2.) * (2. * tita / pi - 1.)
```

Figura 4: Código del ángulo de la función 'max_left_shoulder_rotation'.

En resumen, dado un vector v de S^2, consideramos $\theta \in [0,\pi]$ y $\varphi \in [-\pi,\pi)$ tales que $v = (\sin\theta\cos\varphi, \cos\theta, \sin\theta\sin\varphi)$ en la base del hombro izquierdo. Se tiene entonces que

$$\psi_I(v) = \begin{cases} \pi/4 & \text{si} \quad \theta = 0\,, \\ 0 & \text{si} \quad \theta = \pi\,, \\ \left(\varphi + \frac{\pi}{4}\right)\left(\frac{2}{\pi}\theta - 1\right) & \text{si} \quad \theta \in (0,\pi/2] \ \text{ y } \varphi \in [-\pi, 0]\,, \\ \left(\varphi + \frac{\pi}{2}\right)\left(\frac{2}{\pi}\theta - 1\right) & \text{si} \quad \theta \in [\pi/2,\pi] \ \text{ y } \varphi \in [-\pi, 0]\,, \\ -\varphi + \frac{\pi}{4}\left(\frac{2}{\pi}\theta - 1\right) & \text{si} \quad \theta \in (0,\pi/2] \ \text{ y } \varphi \in [0, \pi)\,, \\ -\varphi + \left(2\varphi + \frac{\pi}{2}\right)\left(\frac{2}{\pi}\theta - 1\right) & \text{si} \quad \theta \in (0,\pi/2] \ \text{ y } \varphi \in [0, \pi)\,. \end{cases}$$

En la Figura 4 podemos ver el código usado para calcular el ángulo en la función 'max_left_shoulder_rotation'.

Hasta aquí el brazo izquierdo. Para el derecho, tenemos que tener en cuenta que tercer vector de la base del hombro apunta hacia el frente, y no hacia atrás. Por otra parte, el ángulo lo consideraremos en sentido horario respecto a la dirección del hombro, justo al contrario que para el brazo izquierdo.

Sea $\psi_D : S^2 \to \mathbb{R}$ la función que asigna, a cada dirección del hombro derecho, el ángulo que estamos considerando. Las estimaciones de las que partimos son equivalentes a las de ψ_I:

$$\begin{cases} \psi_D(0,1,0) &= \pi/4\,, \\ \psi_D(\cos\varphi, 0, \sin\varphi) &= 0\,, \text{ para cada } \varphi \in [0,\pi]\,, \\ \psi_D(0,0,-1) &= -\pi/2\,, \\ \psi_D(0,-1,0) &= 0\,. \end{cases}$$

Y, siguiendo las ideas anteriores, llegamos a que, si v es un vector de S^2, y consideramos $\theta \in [0,\pi]$ y $\varphi \in (-\pi,\pi]$ tales que $v = (\sin\theta\cos\varphi, \cos\theta, \sin\theta\sin\varphi)$ en la base del hombro derecho, se tiene entonces que

$$\psi_D(v) = \begin{cases} \pi/4 & \text{si} \quad \theta = 0\,, \\ 0 & \text{si} \quad \theta = \pi\,, \\ \left(\frac{\pi}{4} - \varphi\right)\left(\frac{2}{\pi}\theta - 1\right) & \text{si} \quad \theta \in (0,\pi/2] \ \text{ y } \varphi \in [0,\pi]\,, \\ \left(\frac{\pi}{2} - \varphi\right)\left(\frac{2}{\pi}\theta - 1\right) & \text{si} \quad \theta \in [\pi/2,\pi] \ \text{ y } \varphi \in [0,\pi]\,, \\ \varphi + \frac{\pi}{4}\left(\frac{2}{\pi}\theta - 1\right) & \text{si} \quad \theta \in (0,\pi/2] \ \text{ y } \varphi \in (-\pi,0]\,, \\ \varphi + \left(\frac{\pi}{2} - 2\varphi\right)\left(\frac{2}{\pi}\theta - 1\right) & \text{si} \quad \theta \in (0,\pi/2] \ \text{ y } \varphi \in (-\pi,0]\,. \end{cases}$$

Es evidente que en estos razonamientos la cantidad de suposiciones no triviales han sido inmensas, así que es natural hacerse las siguientes preguntas: ¿Por qué? ¿Por qué esto debería funcionar? En primer lugar, por ahora realmente no sé si esto debería funcionar, aunque creo y espero que sí. En segundo lugar, debemos tener claro que el objetivo de estas funciones es ser aplicadas para el estudio del movimiento haciendo lengua de signos, por lo que las posiciones de los brazos no deberían estar en los límites del cuerpo humano. Probablemente, si aplicáramos estas funciones para estudiar los movimientos de una persona haciendo yoga, el resultado sería desastroso. La idea tras estas funciones es considerar, para cada dirección del hombro, una rotación de hombro de referencia, y esperar que ninguna de las posiciones que nos encontremos en la lengua de signos sobrepase esta rotación o justamente la opuesta. En resumen, no hay ninguna garantía de que estas funciones vayan a servir, pero considero que hay motivos por los que pensar que no deberían dar problemas.

Ya hemos completado los dos primeros pasos, los dos siguientes son prácticamente inmediatos. En primer lugar, vamos a calcular el vector 'aux_normal' como el vector normal al plano que contiene a la dirección del hombro y al vector auxiliar del primer paso. Teniendo en cuenta el sentido del ángulo que hemos considerado en el segundo paso (antihorario para el brazo izquierdo, horario para el brazo derecho), el vector 'aux_normal' va a ser el producto tensorial del vector auxiliar con la dirección del hombro para el brazo derecho, y el mismo producto pero en el otro orden para el brazo derecho. Una vez que tenemos este vector, para obtener la dirección del antebrazo cuando el ángulo del codo es de 90 grados y el hombro está rotado al máximo es símplemente

$$\cos(\text{angulo})\,\text{aux\_vector} + \sin(\text{angulo})\,\text{aux\_normal}\,.$$

El último paso consiste símplemente en hacer el producto tensorial entre la dirección del hombro y el vector recién calculado y devolver el resultado. En la Figura 5 vemos las líneas de la función 'max_left_shoulder_rotation'que realizan estos dos últimos pasos.

5. pose2arm_position

El objetivo de esta función es traducir de 'half_pose_landmarks' a 'Atributos'. Así, los parámetros de entrada de la función son un array de NumPy

```
aux_normal = np.cross(aux_vector, left_shoulder_direction)
left_elbow_direction_max_rotation = np.cos(angle) * aux_vector + np.sin(angle) * aux_normal

return np.cross(left_shoulder_direction, left_elbow_direction_max_rotation)
```

Figura 5: Últimas líneas de la función 'max_left_shoulder_rotation'.

de dimensión $(14, 3)$ ó $(14, 4)$ que represente la información de los landmarks asociados al tronco y brazos, y el ancho y alto de la imagen de la que se han obtenido los landmarks. La función devuelve un diccionario con todos los atributos de los brazos.

Lo primero que hacemos es desnormalizar los landmarks, para que todos los ejes estén en la misma escala. Para ello, definimos el vector de escala como un vector de tres componentes en las que la primera y la última son el ancho de la imagen y la segunda el alto. A continuación, multiplicamos el array 'half_pose_landmarks' por el vector de escala.

Empezamos ya con el brazo izquierdo. El primer paso es definir la base del hombro que vemos en la Figura 1. Para ello, inicializamos una array de dimensión $(3, 3)$. Almacenaremos cada vector de la base en una columna del array. El primer vector tiene la dirección y el sentido del vector que va del hombro derecho al hombro izquierdo. A continuación, calculamos la dirección (y el sentido) del tercer vector haciendo el producto vectorial del primer vector con el vector que va del hombro izquierdo a la cadera izquierda. Este vector es ortogonal al plano generado por los hombros y la cadera izquierda y apunta hacia atrás. Para calcular la dirección y el sentido del segundo vector, basta con hacer el producto vectorial del tercer vector con el primero. Lo único que falta para tener una base ortonormal es dividir cada vector por su norma

Antes de seguir, es conveniente hacer varios comentarios. En primer lugar, llamaremos "base de la imagen" a la base en la que están los landmarks. En segundo lugar, a la base que acabamos de construir le llamamos "base del hombro izquierdo". También diremos que el array que acabamos de definir, 'left_shoulder_basis' en el código, es la matriz de cambio de base de la base de la imagen a la base del hombro izquierdo. Esto se debe a que, si consideramos un array v de dimensión $(3,)$ representando a un vector en la base de la imagen, entonces el array v@left_shoulder_basis representa al mismo vector pero en la base del hombro izquierdo (@ denota al producto matricial, tal y como pasa en NumPy). Por último, como la base del hombro izquierdo es ortonormal, la matriz inversa de 'left_shoulder_basis' es precisamente su traspuesta. Por tanto, diremos que el array 'left_shoulder_basis.T' es la matriz de cambio de base de la base del hombro izquierdo a la base de la imagen.

El siguiente paso consiste en obtener la dirección del hombro izquierdo en la base del hombro izquierdo y la longitud del brazo (el brazo es lo que hay entre el hombro y el codo). Para ello, primero obtenemos el vector del brazo en la base de la imagen, calculamos su norma (longitud del brazo) y dividimos el vector entre la norma y le cambiamos la base. En la Figura 6 tenemos las líneas de código que realizan los tres pasos que llevamos.

```
scale = np.array([frame_width, frame_height, frame_width], dtype=np.float64)
half_pose_landmarks_ = half_pose_landmarks[:, :3] * scale

left_shoulder_basis = np.empty((3, 3), dtype=np.float64)
left_shoulder_basis[:, 0] = half_pose_landmarks_[0] - half_pose_landmarks_[1]
left_shoulder_basis[:, 2] = np.cross(left_shoulder_basis[:, 0], half_pose_landmarks_[12] - half_pose_landmarks_[0])
left_shoulder_basis[:, 1] = np.cross(left_shoulder_basis[:, 2], left_shoulder_basis[:, 0])
left_shoulder_basis /= np.linalg.norm(left_shoulder_basis, axis=0, keepdims=True)

left_upperarm_vector = half_pose_landmarks_[2] - half_pose_landmarks_[0]
left_upperarm_length = np.linalg.norm(left_upperarm_vector)
left_shoulder_direction = (left_upperarm_vector / left_upperarm_length) @ left_shoulder_basis
```

Figura 6: Primeras líneas de código de la función 'pose2arm_position'.

```
left_forearm_vector = half_pose_landmarks_[4] - half_pose_landmarks_[2]
left_forearm_length = np.linalg.norm(left_forearm_vector)
left_forearm_direction = left_forearm_vector / left_forearm_length
left_elbow_angle = np.pi - np.arccos(np.clip(np.dot(left_upperarm_vector, left_forearm_direction) / left_upperarm_length, -1, 1))

if np.isclose(left_elbow_angle, np.pi, atol=1e-3):
    left_elbow_angle = np.float64(np.pi)
    left_shoulder_rotation = np.float64(0)
else:
    left_elbow_direction = left_forearm_direction @ left_shoulder_basis
    left_arm_plane_normal_vector = np.cross(left_shoulder_direction, left_elbow_direction)
    left_arm_plane_normal_vector /= np.linalg.norm(left_arm_plane_normal_vector)
    left_arm_max_rotation_plane_normal_vector = max_left_shoulder_rotation(left_shoulder_direction)
    left_shoulder_rotation = np.arccos(np.clip(np.dot(left_arm_plane_normal_vector, left_arm_max_rotation_plane_normal_vector), -1, 1))
```

Figura 7: Rotación del hombro, ángulo del codo y longitud del antebrazo.

A continuación, obtendremos la rotación del hombro, el ángulo del codo y la longitud del antebrazo. Lo hacemos en el orden inverso. Para obtener la longitud del antebrazo basta con calcular la norma de su vector. Para calcular el ángulo que forma el codo, lo único que hay que hacer es restar a π el ángulo entre los vectores asociados al brazo y al antebrazo. Para la rotación del hombro sí que tenemos que trabajar un poco más.

Por una parte, la rotación del hombro solo la podemos estimar (por ahora) si el codo no está totalmente extendido. Así, en caso de que el ángulo del codo sea próximo a π, por una parte fijamos el ángulo del codo a π y, por otra parte, por el momento suponemos que el hombro está rotado al máximo (luego lo ajustaremos teniendo en cuenta la mano).

Por otra parte, si el codo no está totalmente extendido, lo que haremos será calcular el ángulo entre el vector unitario y normal al brazo y antebrazo izquierdos cuando el hombro está en la dirección actual y totalmente rotado y el vector unitario normal al plano que contiene el brazo y antebrazo izquierdos. Para ello, primero expresamos el vector del antebrazo en la base del hombro. Segundo, obtenemos un vector unitario normal al plano que contiene el brazo y antebrazo izquierdo haciendo el producto vectorial entre la dirección del hombro y la dirección del antebrazo y diviendo el resultado por su norma. Tercero, obtenemos el vector unitario y normal al brazo y antebrazo izquierdos cuando el hombro está en la dirección actual y totalmente rotado aplicando la función 'max_left_shoulder_rotation'. Por último, calculamos el ángulo entre los dos vectores normales calculando el arcocoseno de su producto escalar. En la Figura 7 tenemos el código que realiza estos cálculos.

```
left_hand_basis = np.empty((3, 3), dtype=np.float64)
left_hand_basis[:, 0] = np.cross(half_pose_landmarks_[6] - half_pose_landmarks_[4], half_pose_landmarks_[8] - half_pose_landmarks_[4])
left_hand_basis[:, 2] = (half_pose_landmarks_[6] + half_pose_landmarks_[8])/2. - half_pose_landmarks_[4]
left_hand_basis[:, 1] = np.cross(left_hand_basis[:, 2], left_hand_basis[:, 0])
left_hand_basis /= np.linalg.norm(left_hand_basis, axis=0, keepdims=True)

left_pinky_vector = (half_pose_landmarks_[6] - half_pose_landmarks_[4]) @ left_hand_basis
left_index_vector = (half_pose_landmarks_[8] - half_pose_landmarks_[4]) @ left_hand_basis
left_thumb_vector = (half_pose_landmarks_[10] - half_pose_landmarks_[4]) @ left_hand_basis

left_wrist_direction = - left_forearm_direction @ left_hand_basis
left_wrist_rotation = azimuth(*(left_wrist_direction * np.array([1, 1, -1], dtype=np.float64)))
left_wrist_inclination = colatitude(*(left_wrist_direction * np.array([1, 1, -1], dtype=np.float64)))
```

Figura 8: Base de la mano, dedos y rotación e inclinación de la muñeca

Ya solo queda la rotación del codo, la rotación e inclinación de la muñeca y los vectores de los dedos. Para estos atributos necesitaremos primero definir la base de la mano. Al igual que con la base del hombro, realmente lo que definimos es la matriz de cambio de base de la base de la imagen a la base de la mano. Así, en primer lugar, inicializamos un array de NumPy de dimensión $(3, 3)$. La primera columna de la matriz representa al vector unitario que sale de la palma perpendicularmente, por lo que su dirección y sentido serán los de el producto vectorial entre el vector que va de la muñeca al nacimiento del meñique y el vector que va de la muñeca al nacimiento del índice. Por otra parte, la tercera columna de la matriz representa al vector unitario que sale de la muñeca en dirección al centro de la palma. En nuestro caso, supondremos que tiene la misma dirección y sentido que el vector que sale de la muñeca y va al punto medio entre el nacimiento del índice y el nacimiento del meñique. Para completar la base ortonormal, la segunda columna de la matriz ha de ser el producto vectorial entre la tercera columna y la primera.

Empecemos ahora por los atributos más sencillos, los vectores de los dedos en la base de la mano. Basta con calcular los vectores que salen de la muñeca y van a los nacimientos de los dedos correspondientes y expresarlos en la base de la mano.

Continuamos con la rotación y la inclinación de la muñeca. Tal y como se comentaba más arriba, estas son las coordenadas esféricas del vector unitario que representa al antebrazo en la base de la mano. Más concretamente, expresaremos el opuesto del vector unitario asociado al antebrazo en la base de la mano, cambiaremos el signo a la tercera componente y obtendremos las coordenadas esféricas asociadas. En la Figura 8 vemos el código que realiza estos últimos pasos.

Ya solo queda la rotación del codo. La idea es comparar la posición de la mano si la muñeca no estuviese inclinada con la posición de la mano si la muñeca no estuviese inclinada y el codo no estuviese rotado. Más específicamente, lo que compararemos serán los vectores normales a la palma de la mano. Por tanto, el primer paso consiste en calcular el vector normal a la palma de la mano si le quitamos la inclinación a la muñeca.

¿Cómo expresamos matemáticamente “quitarle la inclinación a la muñeca”? En primer lugar, claramente estamos hablando de una rotación, puesto que lleva la base de la mano a otra base (la de la mano en caso de que la muñeca

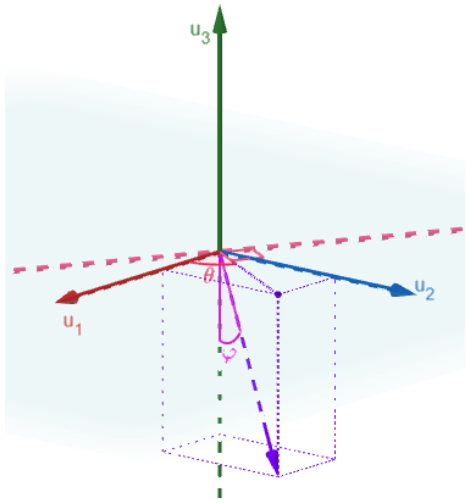

Figura 9: Representación gráfica del eje de rotación.

no estuviese inclinada). Ahora, pensando en cómo actúa esta rotación sobre el tercer vector de la base de la mano, que lo lleva al vector unitario que representa a la dirección del antebrazo, podemos entender que se trata de una rotación del mismo ángulo que la inclinación de la muñeca y respecto al vector del plano formado por los dos primeros vectores de la base y perpendicular a la proyección del antebrazo sobre el plano formado por los dos primeros vectores de la base. En la Figura 9 podemos ver una representación gráfica de la base de la mano, el vector unitario opuesto al antebrazo, la rotación e inclinación de la muñeca y el eje de la rotación que le quita la inclinación a la muñeca.

En definitiva, si le quitamos la inclinación a la muñeca, el vector normal a la palma de la mano en la base de la mano es el vector resultante de aplicarle al vector $(1,0,0)$ la rotación de φ radianes (en sentido antihorario) respecto al vector $(\sin\theta, -\cos\theta, 0)$, donde φ y θ son la inclinación y la rotación de la muñeca, respectivamente.

Una forma de calcular rotaciones es mediante la Fórmula de rotación de Rodrigues. Esta fórmula dice que, siendo v un vector de $\mathbb{R}^3$, r un vector unitario de $\mathbb{R}^3$ y φ un número real, el resultado de aplicar a v una rotación respecto r de ángulo φ en sentido antihorario es $v' = v_{\parallel} + \cos\varphi \cdot v_{\perp} + \sin\varphi \cdot (r \otimes v)$, donde $v_{\parallel} = (v \cdot r) \cdot r$ y $v_{\perp} = v - v_{\parallel}$. En nuestro caso, $v = (1,0,0)$ y $r = (\sin\theta, -\cos\theta, 0)$, por lo que $v_{\parallel} = \left(\sin^2\theta, -\sin\theta\cos\theta, 0\right)$, $v_{\perp} = \left(\cos^2\theta, \sin\theta\cos\theta, 0\right)$ y $r \otimes v = (0, 0, \cos\theta)$. Por tanto, el vector normal a la palma de la mano si la muñeca no

estuviese inclinada es:

$$\begin{aligned} v' &= v_{\parallel} + \cos\varphi \cdot v_{\perp} + \sin\varphi \cdot (r \otimes v) \\ &= \left(\sin^2\theta + \cos\varphi\cos^2\theta, -\sin\theta\cos\theta + \cos\varphi\sin\theta\cos\theta, \sin\varphi\cos\theta\right) \\ &= \left(1 + (\cos\varphi - 1)\cos^2\theta, (\cos\varphi - 1)\sin\theta\cos\theta, \sin\varphi\cos\theta\right). \end{aligned}$$

Ahora debemos diferenciar entre dos situaciones: si tenemos el codo totalmente extendido o no. Empecemos suponiendo que no. En este caso, ya hemos calculado el vector normal al plano que contiene tanto el brazo como el antebrazo. Lo que supongo ahora es que, en caso de que el codo no esté rotado y la muñeca no esté inclinada, el vector normal a la palma de la mano es, precisamente, el vector normal al plano que contiene tanto el brazo como el antebrazo. Y, además, considero cierto que el codo se puede rotar solo 90 grados en cada dirección. Teniendo esto en cuenta, para calcular la rotación del hombro basta con expresar el vector normal al plano que contiene tanto el brazo como el antebrazo en la base de la mano y calcular el ángulo que forma el vector normal a la palma de la mano si la muñeca no estuviese inclinada. Cabe decir que al rotar el antebrazo en sentido horario el ángulo crecería, mientras que al rotarlo en sentido antihorario el ángulo decrecería.

Supongamos ahora que el codo está totalmente extendido. En este caso, todavía no hemos estimado la rotación del hombro y existen infinitos planos distintos que contienen tanto al brazo como al antebrazo, por lo que la estrategia debe ser distinta a la anterior. Así pues, a diferencia del caso anterior, en el que tomábamos como referencia la palma de la mano cuando el hombro estaba rotado según nuestros cálculos y el codo no estaba rotado, ahora vamos a tomar como referencia la palma de la mano cuando el brazo está totalmente rotado (tanto el hombro como el codo). De nuevo, supongo que, cuando el brazo está totalmente rotado y la muñeca no está inclinada, entonces lo que sería en ese caso el segundo vector de la base de la mano es justamente el vector que resulta de aplicar la función 'max_left_shoulder_rotation' a la dirección del hombro.

Entonces, para obtener el vector normal a la palma de la mano cuando el brazo está totalmente rotado y la muñeca no está inclinada multiplicamos el vector opuesto a la dirección del antebrazo por el resultado de aplicar la función 'max_left_shoulder_rotation' a la dirección del hombro, todo en la base de la mano. Ahora, para obtener la rotación total del brazo (hombro más codo), calculamos el ángulo entre el vector que acabamos de obtener y el vector normal a la palma de la mano cuando la muñeca no está inclinada. Para ello, debemos tener en cuenta que, en este caso, el rango de rotación es de una vuelta completa (media del hombro y media del codo), por lo que debemos tener cuidado en diferenciar los ángulos de menos de media vuelta con los de más. Por último, repartiremos equitativamente la rotación entre el hombro y el codo, teniendo en cuenta que el hombro rota entre 0 y π radianes y el codo entre $-\pi/2$ y $\pi/2$. En la Figura 10 tenemos el código para estimar la rotación del codo.

Ya hemos visto cómo obtenemos los atributos del brazo izquierdo. Para obtener los del brazo derecho el proceso es análogo, pero debemos hacer algún cambio

```
a = np.cos(left_wrist_inclination)
b = np.sin(left_wrist_inclination)
c = np.cos(left_wrist_rotation)
d = np.sin(left_wrist_rotation)
left_palm_normal_vector_no_wrist_inclination = np.array([1.+(a-1.)*c*c, (a-1.)*c*d, b*c], dtype=np.float64)

if np.isclose(left_elbow_angle, np.pi):
    aux_vector = max_left_shoulder_rotation(left_shoulder_direction) @ left_shoulder_basis.T @ left_hand_basis
    left_palm_normal_vector_max_arm_rotation = np.cross(left_wrist_direction, aux_vector)
    if np.dot(left_palm_normal_vector_no_wrist_inclination, aux_vector)>=0:
        left_arm_rotation = np.arccos(np.clip(
            np.dot(left_palm_normal_vector_no_wrist_inclination, left_palm_normal_vector_max_arm_rotation),
            -1, 1))
    else:
        left_arm_rotation = np.float64(2.*np.pi) - np.arccos(np.clip(
            np.dot(left_palm_normal_vector_no_wrist_inclination, left_palm_normal_vector_max_arm_rotation),
            -1, 1))
    left_shoulder_rotation = left_arm_rotation / 2.
    left_elbow_rotation = left_arm_rotation - left_shoulder_rotation - np.float64(np.pi/2.)

else:
    left_palm_normal_vector_no_elbow_rotation = left_arm_plane_normal_vector @ left_shoulder_basis.T @ left_hand_basis
    aux_vector = np.cross(left_palm_normal_vector_no_elbow_rotation, left_wrist_direction)
    aux_vector = np.cross(left_palm_normal_vector_no_elbow_rotation, left_wrist_direction)
    left_elbow_rotation = np.sign(np.dot(left_palm_normal_vector_no_wrist_inclination, aux_vector)) * np.arccos(np.clip(
        np.dot(left_palm_normal_vector_no_wrist_inclination, left_palm_normal_vector_no_elbow_rotation), -1, 1))
```

Figura 10: Código para estimar la rotación del codo.

de signo en los lugares apropiados para que todo cuadre. Recordemos que el tercer vector de la base del hombro derecho debe apuntar hacia el frente para mantener la orientación. De igual forma, el segundo vector de la base de la mano derecha apunta hacia el lado del pulgar, en vez de hacia el lado del meñique. Además, el sentido positivo de la rotación del brazo derecho es el antihorario. Y, también, los vectores devueltos por la función 'max_right_shoulder_rotation' "apuntan en la misma dirección" que los devueltos por 'max_left_shoulder_rotation'.

Todas estas consideraciones hacen que debamos cambiar el orden de algunos productos vectoriales (por ejemplo, para obtener el tercer vector de la base de la mano), que debamos cambiar el signo de algún vector (por ejemplo, cuando obtenemos el vector normal a la palma si el codo no está rotado y la muñeca no está inclinada, en el caso de que el codo no esté totalmente extendido), o que debamos cambiar el sentido de alguna desigualdad (cuando obtener la rotación total del brazo en el caso de que el codo esté totalmente extendido).

6. arm_position2pose

Esta función pretende ser la inversa de 'pose2arm_position', es decir, su objetivo es traducir de 'Atributos' a 'half_pose_landmarks'. Para ello, partirá de un diccionario de atributos de los brazos, de los landmarks del tronco (hombros y cintura) y de la anchura y altura de la imagen en píxeles.

Al igual que en la función anterior, lo primero que hacemos es definir el array de escala. A continuación inicializamos el array que contendrá las 'half_pose_landmarks' sin escalar y rellenamos las filas asociadas al tronco reescalando los landmarks del tronco.

En lo que sigue, el objetivo es rehacer los pasos de la función anterior. Así,

```
scale = np.array([frame_width, frame_height, frame_width], dtype=np.float64)
half_pose_landmarks = np.empty((14, 3), dtype=np.float64)
half_pose_landmarks[[0, 1, 12, 13]] = trunk_landmarks * scale


left_shoulder_basis = np.empty((3, 3), dtype=np.float64)
left_shoulder_basis[:, 0] = half_pose_landmarks[0] - half_pose_landmarks[1]
left_shoulder_basis[:, 2] = np.cross(left_shoulder_basis[:, 0], half_pose_landmarks[12] - half_pose_landmarks[0])
left_shoulder_basis[:, 1] = np.cross(left_shoulder_basis[:, 2], left_shoulder_basis[:, 0])
left_shoulder_basis /= np.linalg.norm(left_shoulder_basis, axis=0, keepdims=True)

left_shoulder_direction_image_basis = arms_position_dict["left_shoulder_direction"] @ left_shoulder_basis.T
half_pose_landmarks[2] = arms_position_dict["left_upperarm_length"] * left_shoulder_direction_image_basis + half_pose_landmarks[0]
```

Figura 11: Primeras líneas de 'arm_position2pose'.

empezamos con el brazo izquierdo. Como tenemos los landmarks del tronco, podemos construir la matriz de cambio de base de la base de la imagen a la base del hombro izquierdo. A continuación, cambiando a la base de la imagen la dirección del hombro y utilizando la longitud del brazo, conseguimos fácilmente el landmark del codo izquierdo. En la Figura 11 vemos el código que realiza estos cálculos.

El siguiente objetivo es encontrar la muñeca izquierda. En el caso de que el codo esté totalmente extendido, resulta fácil encontrarla puesto que la dirección del antebrazo y la del hombro coinciden. En caso contrario, tenemos que usar la rotación del hombro izquierdo y el ángulo del codo para calcular la dirección del antebrazo y así encontrar la muñeca.

Los pasos a seguir son los siguientes: en primer lugar, obtenemos el vector normal al plano del brazo cuando el hombro está totalmente rotado; en segundo lugar, consideramos el vector auxiliar que surge de hacer el producto vectorial de la dirección del hombro con el vector normal que acabamos de calcular; en tercer lugar, obtenemos el vector normal al plano que contiene el brazo rotando el vector normal al plano del brazo cuando el hombro está totalmente rotado un ángulo igual a la rotación del hombro en dirección al vector auxiliar; en cuarto lugar, consideramos un nuevo vector auxiliar que surge de hacer el producto vectorial del vector normal al plano del brazo con la dirección del hombro (este vector es la dirección del antebrazo en el caso de que el ángulo del codo sea de 90 grados); por último, obtenemos la dirección del antebrazo rotando el vector opuesto a la dirección del hombro un ángulo igual al ángulo del codo en dirección al nuevo vector auxiliar. Una vez encontrada la dirección del antebrazo, a partir del landmark del codo y de la longitud del antebrazo encontramos fácilmente la muñeca izquierda. En la Figura 12 vemos el código para encontrar el landmark de la muñeca.

Ya solo nos quedan por determinar los landmarks de los nacimientos de los dedos. Recordemos que en el diccionario de atributos lo que tenemos son los vectores asociados en la base de la mano. Por supuesto, teniendo el landmark de la muñeca y los vectores de los dedos, dar los landmarks es inmediato, sin embargo, resulta crucial encontrar la base de la mano. Así pues, nuestro objetivo es calcular cada vector de la base de la mano izquierda. Para ello, primero encontraremos la base de la mano en el caso de que la muñeca no estuviese inclinada, y, seguidamente, aplicaremos la rotación adecuada a esta base, teniendo

```
if np.isclose(arms_position_dict["left_elbow_angle"], np.pi):
    half_pose_landmarks[4] = arms_position_dict["left_forearm_length"] * left_shoulder_direction_image_basis + half_pose_landmarks[2]
else:
    left_arm_max_rotation_plane_normal_vector = max_left_shoulder_rotation(arms_position_dict["left_shoulder_direction"])
    aux_vector = np.cross(arms_position_dict["left_shoulder_direction"], left_arm_max_rotation_plane_normal_vector)
    left_arm_plane_normal_vector = (np.cos(arms_position_dict["left_shoulder_rotation"]) * left_arm_max_rotation_plane_normal_vector +
                                    np.sin(arms_position_dict["left_shoulder_rotation"]) * aux_vector)

    aux_vector = np.cross(left_arm_plane_normal_vector, arms_position_dict["left_shoulder_direction"])
    left_forearm_direction = (- np.cos(arms_position_dict["left_elbow_angle"]) * arms_position_dict["left_shoulder_direction"] +
                              np.sin(arms_position_dict["left_elbow_angle"]) * aux_vector) @ left_shoulder_basis.T
    half_pose_landmarks[4] = arms_position_dict["left_forearm_length"] * left_forearm_direction + half_pose_landmarks[2]
```

Figura 12: Código para obtener el landmark de la muñeca izquierda.

en cuenta la rotación e inclinación de la muñeca.

Si la muñeca no estuviese inclinada, el tercer vector de la base de la mano es precisamente la dirección del antebrazo. Como la base de la mano es ortonormal y positivamente orientada, basta con que encontremos otro vector de la misma para determinar también el que falte. Evidentemente, vamos a calcular el primer vector de la base, es decir, el vector normal a la palma de la mano en caso de que la muñeca no esté inclinada.

En caso de que el codo esté totalmente extendido, el primer paso que debemos hacer es calcular la rotación total del brazo. Tal y como la habíamos definido en la función anterior, para ello debemos sumar las rotaciones del hombro y del codo y añadir otros $\pi/2$. Ahora, recordemos que, cuando el brazo estaba totalmente rotado, suponíamos que el vector que resulta de aplicar la función 'max_left_shoulder_rotation' a la dirección del hombro es precisamente el segundo vector de la base de la mano cuando la muñeca no está inclinada. Consideremos como vector auxiliar este vector en la base de la imagen. Entonces, el vector normal a la palma de la mano cuando el hombro está totalmente rotado y la muñeca no está inclinada es el producto vectorial de este vector auxiliar con la dirección del antebrazo. Para conseguir el vector normal a la palma de la mano en el caso de que la muñeca no esté inclinada, lo único que falta hacer es rotar el vector normal a la palma de la mano cuando el hombro está totalmente rotado y la muñeca no está inclinada un ángulo igual a la rotación del brazo en dirección al vector auxiliar.

En caso de que el codo no esté totalmente extendido, suponíamos que el vector normal a la palma de la mano cuando la muñeca no estaba inclinada y el codo no estaba rotado era precisamente el vector normal al plano que contiene tanto el brazo como el antebrazo. Consideramos ahora el vector auxiliar que surge de hacer el producto vectorial de la dirección del antebrazo con el vector normal a la palma de la mano cuando el codo no está rotado y, en definitiva, el vector normal a la palma si la muñeca no está inclinada es el resultado de rotar el vector normal a la palma cuando el codo no está rotado un ángulo igual a la rotación del codo en dirección al vector auxiliar.

Ya tenemos pues el vector normal a la palma cuando la muñeca no está inclinada, y por tanto la base de la mano en el mismo supuesto. Para conseguir la base de la mano teniendo en cuenta la rotación e inclinación de la muñeca, basta con que realicemos la rotación inversa a la que realizamos en la función

'pose2arm_position'. Para ello, conviene tener en cuenta el siguiente resultado trivial:

Proposición 1. *Sean $\mathcal{U}$ una base de $\mathbb{R}^3$ y r un vector de $\mathbb{R}^3$. Sea $\mathcal{V}$ la base de $\mathbb{R}^3$ que surge de rotar $\mathcal{U}$ respecto al eje r un ángulo cualquiera. Entonces, las coordenadas de r en las bases $\mathcal{U}$ y $\mathcal{V}$ coinciden.*

Demostración. Llamemos R a la rotación, u_1, u_2, u_3 a los vectores de $\mathcal{U}$, v_1, v_2, v_3 a los vectores de $\mathcal{V}$ y sean (r_1, r_2, r_3) las coordenadas de r en la base $\mathcal{U}$.

Obviamente, $Rr = r$. Por otra parte, como R es lineal,

$$Rr = R\left(r_1u_1 + r_2u_2 + r_3u_3\right) = r_1v_1 + r_2v_2 + r_3v_3\,.$$

En definitiva, (r_1, r_2, r_3) son las coordenadas de r en la base $\mathcal{V}$. □

En la función 'pose2arm_position', para calcular el vector normal a la palma de la mano si la muñeca no estuviese inclinada lo que hacíamos era rotar el primer vector de la base de la mano un ángulo igual a la inclinación de la muñeca respecto al vector $(\sin\theta, -\cos\theta, 0)$ en la base de la mano, donde θ era la rotación de la muñeca. La rotación inversa es, por supuesto, la que surge de rotar un ángulo igual al opuesto de la inclinación de la muñeca respecto al mismo eje. De hecho, es claro que si rotamos toda la base de la mano mediante esta rotación, obtenemos la base que tenemos ahora nosotros.

Por la Proposición 1, las coordenadas de este vector en nuestra base son también $(\sin\theta, -\cos\theta, 0)$, con θ la rotación de la muñeca. Así, para obtener la base de la mano izquierda debemos rotar cada vector de nuestra base un ángulo opuesto a la inclinación de la muñeca respecto al vector mencionado. Llamando φ a la inclinación de la muñeca y usando la Fórmula de Rodrigues, aplicando esta rotación a un vector v obtendríamos

$$v' = v_{\parallel} + \cos\left(-\varphi\right)\cdot v_{\perp} + \sin\left(-\varphi\right)\cdot\left(r\otimes v\right) = v_{\parallel} + \cos\varphi\cdot v_{\perp} - \sin\varphi\cdot\left(r\otimes v\right).$$

Si llamamos v_1, v_2 y v_3 a los vectores de la base que hemos calculado y u_1, u_2 y u_3 a los vectores de la base de la mano, haciendo los cálculos se puede comprobar que

$$\begin{aligned}
u_1 &= \left(1 + (\cos\varphi - 1)\cos^2\theta\right)v_1 + (\cos\varphi - 1)\sin\theta\cos\theta\, v_2 - \sin\varphi\cos\theta\, v_3\,,\\
u_2 &= \left(1 + (\cos\varphi - 1)\sin^2\theta\right)v_2 + (\cos\varphi - 1)\sin\theta\cos\theta\, v_1 - \sin\varphi\sin\theta\, v_3\,,\\
u_3 &= \sin\varphi\cos\theta\, v_1 + \sin\varphi\sin\theta\, v_2 + \cos\varphi\, v_3\,.
\end{aligned}$$

Una vez obtenida la base de la mano izquierda, es inmediato obtener los landmarks de los nacimientos de los dedos a partir del landmark de la muñeca y de los vectores correspondientes. En la Figura 13 vemos el código que realiza estos últimos pasos.

Igual que en la función anterior, los cálculos para el brazo derecho son análogos. Bastará con tener cuidado con los signos y orientaciones.

```
if np.isclose(arms_position_dict["left_elbow_angle"], np.pi):
    left_arm_rotation = arms_position_dict["left_shoulder_rotation"] + arms_position_dict["left_elbow_rotation"] + np.float64(np.pi/2.)

    aux_vector = max_left_shoulder_rotation(arms_position_dict["left_shoulder_direction"]) @ left_shoulder_basis.T
    left_palm_normal_vector_max_arm_rotation = np.cross(aux_vector, left_forearm_direction)

    left_palm_normal_vector_no_wrist_inclination = (np.cos(left_arm_rotation) * left_palm_normal_vector_max_arm_rotation +
                                                    np.sin(left_arm_rotation) * aux_vector)
else:
    left_palm_normal_vector_no_elbow_rotation = left_arm_plane_normal_vector @ left_shoulder_basis.T
    aux_vector = np.cross(left_forearm_direction, left_palm_normal_vector_no_elbow_rotation)

    left_palm_normal_vector_no_wrist_inclination = (np.cos(arms_position_dict["left_elbow_rotation"]) * left_palm_normal_vector_no_elbow_rotation +
                                                    np.sin(arms_position_dict["left_elbow_rotation"]) * aux_vector)

v1 = left_palm_normal_vector_no_wrist_inclination
v3 = left_forearm_direction
v2 = np.cross(v3, v1)

a = np.cos(arms_position_dict["left_wrist_inclination"])
b = np.sin(arms_position_dict["left_wrist_inclination"])
c = np.cos(arms_position_dict["left_wrist_rotation"])
d = np.sin(arms_position_dict["left_wrist_rotation"])
left_hand_basis = np.empty((3, 3), dtype=np.float64)
left_hand_basis[:, 0] = (1.+(a-1.)*c*c) * v1 + (a-1.)*c*d * v2 - b*c * v3
left_hand_basis[:, 1] = (1.+(a-1.)*d*d) * v2 + (a-1.)*c*d * v1 - b*d * v3
left_hand_basis[:, 2] = b*c * v1 + b*d * v2 + a * v3

half_pose_landmarks[6] = arms_position_dict["left_pinky_vector"] @ left_hand_basis.T + half_pose_landmarks[4]
half_pose_landmarks[8] = arms_position_dict["left_index_vector"] @ left_hand_basis.T + half_pose_landmarks[4]
half_pose_landmarks[10] = arms_position_dict["left_thumb_vector"] @ left_hand_basis.T + half_pose_landmarks[4]
```

Figura 13: Landmarks de los nacimientos de los dedos de la mano izquierda.