Domotica y Entornos Industriales

DanielAsensiRoch: **48776120C**

ElviMihaiSabauSabau: **51254875L**

VadymFormanyuk: **x5561410x**

EvaSabaterVillora: **46086358Q**

MarcosDiazHellinGarcia: **48796739P**

25 de octubre de 2022

Índice

1. Problema

La problemática planteada en esta práctica, es la realización de un sistema el cual mediante visión por computador sea capaz de reconocer y trackear de manera precisa el movimiento de una mano, de tal manera que sea capaz de discernir entre los diferentes desplazamientos que dicha mano realice siendo estos los siguientes:

- Movimiento horizontal
- Movimiento vertical
- Movimiento circular
- Movimiento frontal en profundidad

Para resolver el problema se ha dividido este en tres fases, siguiendo el enunciado de la práctica y lo explicado en durante las clases prácticas, el RoadMap es el siguiente:

1. Obtención de la sección de interes (ROI)
2. Tracking de la región
3. Análisis de datos

2. Detectar la sección de interés (ROI)

Esta parte del proceso de reconocimiento consiste en substraer el área que nos transmitirá la información. En nuestro caso de análisis nuestro **ROI** será la mano de nuestro sujeto, dependiendo de los movimientos del sujeto la información reconocida será diferente. Lo primero que debemos hacer es aislar el ROI del resto del entorno, para ello usaremos diferente técnicas de abstracción de manera concatenada.

Como hemos podido investigar para la realización de la práctica, lo primero que debemos hacer es una pre-segmentación, la cual se realizará ajustando todos los frames o fotos de la secuencia para que tengan unas características similares, en el caso de la realización de esta práctica los ajustes ya vienen hechos en nuestro set de imágenes, de otra forma algunas de las técnicas aplicables para las imágenes serían:

- Realizar un reescalado de las imagenes para que todas tengan el mismo tamaño
- Ajustar el brillo de todos los frames para que ninguno supere cierto umbral
- Realizar un recorte de la cantidad de colores expuestos en la imagen

2.1. Segmentación de los datos

En este paso aplicaremos los diferentes algoritmos de segmentación disponibles, estos serán aplicados a cada uno de los frames de la imagen para extraer el ROI deseado

- **Colores:** a partir de las diferentes cotas RGB y realizando diferentes pruebas a medida que el código se iba ejecutando hemos acotado los valores para extraer los valores rosados y rojizos propios de una mano humana, en concreto se han utilizado los siguientes valores RGB:
  - **Rojo: 140-250**
  - **Verde: 150-220**
  - **Blue: 100-250**

Figura 1: Colores aplicados

- **Profundidad:** nuestros frames a parte de los valores RGB, tenemos una cuarta dimensión de la matriz a la cual denominamos D **Depth** y representa la profundidad. Con este valor podemos identificar cuales de los píxeles se encuentran más cercanos a la cámara y cuales se encuentran más alejados. En este caso solo nos interesan aquellos que se encuentran más cercanos a la cámara ya que representarán a la mano, para ello deberemos aplicar un umbral para descartar los píxeles más lejanos.

Figura 2: Colores aplicados y profundidad

- **Extracción del fondo:** para extraer el fondo de las imágenes nos ayudaremos de un frame proporcionado por el profesor, con el cual si realizamos la resta en valor absoluto de la imagen menos el fondo o viceversa y el valor del píxel es superior a un umbral,significa que el píxel difiere del fondo y por lo tanto lo añadiremos a nuestro conjunto de análisis.

Figura 3: Colores aplicados, profundidad y extracción de fondo aplicados

Problemas encontrados durante la extracción del background:

- **Tipos de valores:** durante la realización de la resta de las matrices que contienen los valores de profundidad y de color de nuestros frames, nos hemos enfrentado al problema de que matlab casteaba dichos valores resultantes de la resta en *unsigned int*, para ello hemos tenido que realizar un casteo al tipo *int64* ya que este si que posee signo.
- **Colores similares:** Uno de los problemas a los que nos hemos enfrentado durante la realización de la exclusión de los valores, es que estos tenían un valor de luminosidad muy parecido por lo tanto son excluidos, este problema no se ha resuelto por lo que se ha excluido directamente la máscara final.

3. Tracking del ROI

El tracking de nuestra región de interés, se base en la formulación de un algoritmo de trackeo que nos permita seguir el movimiento de nuestro ROI, de manera más o menos acotada a lo largo de la secuencia de frames. Para la realización de este trackeo hemos utilizado los centroides.

Centroides: Centro de masa de la región, devuelto como un vector de 1 por Q, donde Q es la dimensionalidad de la imagen. El primer elemento de Centroid es la coordenada horizontal (o coordenada x) del centro de masa. El segundo elemento es la coordenada vertical (o coordenada y). Todos los otros elementos de Centroid se encuentran en orden de dimensión.

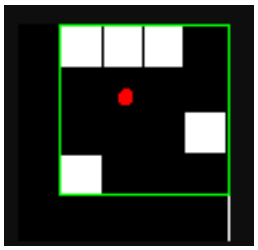

Figura 4: Centroide

Esta figura ilustra el centroide y el cuadro delimitador de una región discontinua. La región está formada por los píxeles blancos. El recuadro verde es el cuadro delimitador y el punto rojo es el centroide.

De esta manera si aplicamos dicha explicación a cada uno de los frames de nuestra secuencia obtenemos un resultado como este:

Figura 5: Centroide

4. Trackeo en matriz de coordenadas

Cuando hemos obtenido absolutamente todos los centroides relevantes dentro de nuestra imagen, deberemos sacar la distancia media entre cada uno de los centroides, representados por puntos en la imagen anterior, una vez obtenidas las distancias medias en los ejes X,Y y Z deberemos realizar la mediana, para eliminar los valores atípicos.

```
if hayCentroides==1
    despX=[];
    despY=[];
    %calculo de distancias: para ello sacamos la media de las profundidades
    %Zi iniciales
    %Zf finales
    Zi=median([profundidad(1) profundidad(2) profundidad(3)]);% coordenada z inicial
    Zf=median([profundidad(end-1) profundidad(end-2) profundidad(end-3)]);% coordenada z final
    despZ=(double(Zi)-double(Zf));

    for i=2:size(centroidesCompletos)
        despY=[despY,abs(centroidesCompletos(i,1)-centroidesCompletos(i-1,1))];
        despX=[despX,abs(centroidesCompletos(i,2)-centroidesCompletos(i-1,2))];
    end

    despX=median(despX);%distancia aumentada en eje x
    despY=median(despY);%distancia aumentada en eje y
    %despZ=median(despZ);
end
```

4.1. Entendimiento de distancias obtenidas

Debemos para poder entender los resultados, volver a describir las situaciones que nuestro sistema de reconocimiento debe de poder comprender:

- **Movimiento horizontal:** este caso se podrá deducir siempre y cuando las distancias varíen en el eje **X**, pero no en el eje **Z** ni en el eje **Y**
- **Movimiento vertical:**este caso se podrá deducir siempre y cuando las distancias varíen en el eje **Y**, pero no en el eje **X** ni en el eje **Z**

- **Movimiento circular:**este caso se podrá deducir siempre y cuando las distancias varíen en el eje **X**,y en el eje **Y** pero no en el eje **Z**
- **Movimiento frontal en profundidad**este caso se podrá deducir siempre y cuando las distancias varíen en el eje **Z**, pero no en el eje **X** ni en el eje **Y**

5. Modelo generado

Para poder reconocer los movimientos de nuestro sujeto, hemos realizado un modelo entrenado con el que poder, mediante la biblioteca de inteligencia artificial de matlab, poder realizar predicciones.

Para poder generar este modelo predictivo, hemos dividido el conjunto de datos etiquetados facilitados por el profesor, en dos conjuntos, **un conjunto de Test (8 secuencias) y un conjunto de entrenamiento (4 secuencias).**

5.1. KNN vecinos

Hemos optado por la utilización del algoritmo KNN vecinos más próximos, el cual es un clasificador de aprendizaje supervisado no paramétrico, que utiliza la proximidad para hacer clasificaciones o predicciones sobre la agrupación de un punto de datos individual. Si bien se puede usar para problemas de regresión o clasificación, generalmente se usa como un algoritmo de clasificación, partiendo de la suposición de que se pueden encontrar puntos similares cerca uno del otro. Para el algoritmo KNN, hemos utilizado un número de 3 vecinos mas cercanos (k = 3) medido por distancias **Cityblock o Manhattan**, hemos elegido esta distancia ya que es la que mayor porcentaje de aciertos nos ha proporcionado, también la euclídea nos daba un porcentaje de acierto similar .

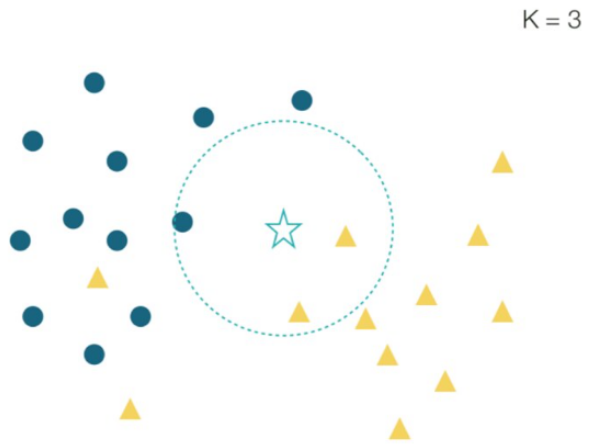

Figura 6: KNN

```
modelo = fitcknn(xtr,ytr,'NumNeighbors',3,'Distance','cityblock');
```

6. Ejecución de código

Para poder ejecutar nuestro código, primero deberemos ejecutar el código contenido en **Clasificador.m**, con este código generaremos el **modelo.mat**. Obteniendo la siguiente salida por terminal:

```
Creando modelo...
Realizando algoritmo
Realizando predicciones...
Acierto del modelo = 100%
Modelo generado con exito
```

Figura 7: Salida terminal

Una vez generado nuestro modelo podremos ejecutar nuestro código main, que se encuentra contenido en archivo **Practical.m**

Cuando dentro del main se hayan analizado todos los frames de la secuencia se procederá a realizar la predicción utilizando el modelo generado anteriormente:

```
datos=[despX,despY,despZ];

%El modelo es el entrenado en el clasificador
resultado = predict(modelo, datos);

switch resultado
    case 1
        disp("Movimiento detect\ado:  Desplazamiento frontal")
    case 2
        disp("Movimiento detectado:  Desplazamiento circular")
    case 3
        disp("Movimiento detectado:  Desplazamiento lateral derecha")
    case 4
        disp("Movimiento detectado:  Desplazamiento hacia arriba")
end
```

```
>> Practical
Cargando secuencia...
Secuencia cargada con exito
Movimiento detectado:  Desplazamiento circular
```

Figura 8: Salida terminal resultados

7. Conclusiones

En conclusión, se ha realizado un sistema eficiente, con el cual se podrían detectar los movimientos de un sujeto siempre y cuando sea de tez clara o rosada, este sistema podría ser fácilmente escalable a más movimientos añadiendo etiquetas a nuestro dataset de datos y resultados, además también podría ser escalable a detectar movimientos en tiempo real, siempre y cuando se le sea suministrada información en forma de matrices de 4 dimensiones.

8. Resultados numéricos

En nuestra implementación contamos con un 100 % de acierto, por lo que la matriz de incertidumbre quedaría de la siguiente manera.

		PREDICCIÓN			
		UP	DOWN	CIRCLE	FORWARD
REAL	UP	1	0	0	0
	DOWN	0	1	0	0
	CIRCLE	0	0	1	0
	FORWARD	0	0	0	1

Figura 9: Resumen resultados numéricos