Domótica y Entornos Inteligentes

Práctica 1: HCI

Francisco Javier Pérez Martínez Francisco Joaquín Murcia Gómez

Óscar David Tremiño Guirao Marcos Cerdán Amat Álex Navarro Soria

5 de noviembre de 2021

Índice

1. Introducción

En esta práctica se busca un sistema capaz de reconocer mediante visión por computador el movimiento de una mano y determinar de que movimiento se trata, ya sea de manera horizontal, vertical, circular o de profundidad.

Para llevar a cabo este sistema, se ha dividido el problema en 3 etapas siguiendo el enunciado de la práctica. En primer lugar, se detectará para cada frame y se segmentará del resto de la escena la región de interés (ROI). Una vez obtenido la región de interés, se implementará un algoritmo de tracking para realizar el seguimiento del ROI a lo largo de la secuencia. Por último, se implementará el análisis de los datos del tracking para extraer información valiosa de la trayectoria.

2. Detección de la región de interés (ROI)

La detección del ROI (Region Of Interest) consiste en substraer el área que nos va a transmitir la información. Para este caso, el ROI será la mano de la persona que esté delante de la cámara. Dependiendo de los movimientos que esta persona haga, la información recibida será diferente. El primer paso para extraer esta información, será aislar el ROI del entorno en el que se encuentre la persona. Como a nosotros solo nos interesa la mano, usaremos distintas técnicas para ignorar el resto del entorno.

La primera fase será la pre-segmentación, la cual ajustará el formato de todos los frames para que tengan unas características más similares y posteriormente sea más fácil trabajar con el conjunto de datos. En nuestro caso, como ya se nos ha dado este preajuste hecho en las capturas, no hemos tenido que hacer nada en este paso, pero algunas de las técnicas que se podrían aplicar a otros datos podrían ser un reescalado de los frames para que todos tengan el mismo tamaño "x y", ajusta el nivel de brillo para que ningún frame supere una cantidad determinada de brillo. En definitiva, este paso se podría considerar un proceso de normalización.

2.1. Segmentación

Después, tendremos que aplicar la fase de segmentación, donde usaremos diferentes algoritmos que aplicaremos a cada frame para extraer el ROI que busquemos. Para nuestro caso, hemos aplicado 2 algoritmos:

- **Color**: a partir de diferentes cotas RGB que nosotros consideremos, seleccionaremos los colores que nosotros queramos incluir en el frame resultado. Por ejemplo, si ponemos que todas las cotas RGB tengan un valor mínimo de 254 y un máximo de 255, en el frame resultante solo quedarán los píxeles de color blanco. Para nuestro caso, hemos puesto una cota de 140-250 para el rojo, una de 150-220 para el verde y una de 100-250 para el azul. Con estas cotas nos aseguraremos de seleccionar los colores rosas-blancos propios de una mano.
- **Profundidad**: las fotos, a parte de unos valores R G B, también incluyen un valor D, depth. Con este valor podemos determinar qué píxeles están más cercanos a cámara y cuáles están más alejados. Para nuestro caso, nos interesa destacar los píxeles más cercanos (los que tengan menos profundidad) a la cámara, puesto que la mano del usuario usualmente será el objeto más cercano a la cámara. Para implementar esto, hemos establecido un threshold. Los píxeles que tengan una profundiad igual o menor a este threshold, se incluirán en el frame final.
- **Background Subtraction**: para quitar el fondo, nos ayudaremos de un frame que consista en el fondo estándar de otras capturas. Este método solo servirá para datos que se mantengan siempre con el mismo fondo, ya que si este varía esta técnica no funcionará. En definitiva, si el valor absoluto de la resta del fondo menos la imagen es superior a un threshold, significa que ese pixel difere del fondo por lo tanto se incluiría en el frame final, de esta forma rescataríamos el frame sin el fondo.

Figura 1: Frame Normal

Figura 2: Frame Segmentado

2.2. Problemas encontrados con el Background Subtraction

- **Resta de matrices defectuosa:** Por defecto, cuando asignaba las correspondientes matrices desde la llamada de la función, matlab les daba un valor de uint, es decir, unsiged int. Esto hacía que, por algún motivo, al hacer la resta del fondo - la imagen, no diesen los valores correctos.

En la anterior foto podemos apreciar como, al hacer la resta de 47 y 175 en valor absoluto tendría que dar 128, sin embargo, en la matriz resultante encontramos un 0. Esto es debido a lo comentado anteriormente sobre los unsigned int.La solución es hacer un casting a int64 directamente desde la llamada.

- **Colores muy similares en ROI:** El algoritmo, resumido de forma muy breve, está hecho para que detecte las zonas en los que los valores RGB cambian entre la imagen y el fondo. Si estos valores, por la luminosidad de la sala o por cualquier motivo son muy parecidos, es posible que se excluya la ROI.

Aquí la mano tendría una zona con un index muy bajo debido a esto que comentaba antes. No he podido encontrar una solución a este problema, por eso he optado por no incluir este método de segmentación en la máscara final.

3. Seguimiento del ROI

3.1. Centroides

El seguimiento del ROI consiste en realizar un algoritmo de tracking que nos permita obtener el seguimiento de la región de interés a lo largo de la secuencia. Para este caso, hemos obtenido los centroides, que no son mas que el punto central de una figura, en nuestro caso la mano que es nuestra región de interés.

Para obtener los centroides, hacemos uso de la función **"regionprops"** donde obtenemos la coordenadas del punto y el área de la región de interés asociada, el problema es que obtenemos en cada frame más de un punto, concretamente un punto por cada área de la región de interés. Para sesgar y obtener el centroide más significativo siempre nos quedamos con el centroide que posea el área mas grande de cada frame obteniendo así una lista de 60 centroides.

Ejemplo de trayectoria obtenida a partir de los centroides seleccionados:

Figura 3: Región de interés

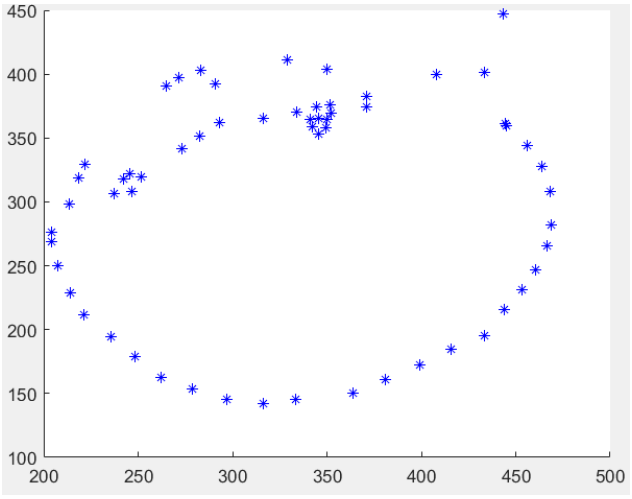

Figura 4: Centroides obtenidos

3.2. Desplazamiento en el eje de coordenadas XYZ

Una vez obtenido estos centroides, debemos de buscar un método que describa la trayectoria que representan estos centroides. El método implementado se basa en el cálculo de la distancia media que hay entre los puntos. Para ello, se han obtenido las distancias relativas de cada eje entre los centroides consecutivos. Una vez sacado las distancias, aplicamos la mediana para eliminar los valores atípicos obteniendo así el valor deseado.

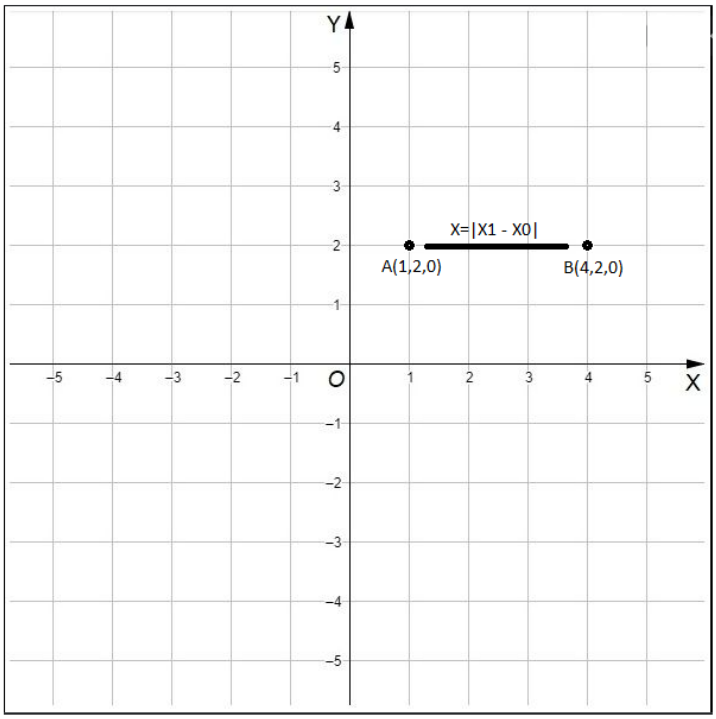

Figura 5: Ejemplo de trayectoria horizontal.

Por ejemplo, si al obtener la diferencia de la coordenada X es aproximadamente 0, en la coordenada Y es superior a 0 podemos determinar que el punto B se encuentra por encima del punto A, formando una trayectoria vertical.
Con este método obtenemos 4 maneras de poder determinar la trayectoria de la mano.

4. Análisis de los datos de la trayectoria

Para poder comenzar el análisis de la trayectoria, hemos identificado 4 situaciones con los datos estudiados en el apartado anterior:

- La primera de ellas seria que se desplaza en el eje X, es decir, que no varia respecto del eje Y ni en el Z pero si respecto del eje X, determinando así un desplazamiento horizontal.
- Si varia respecto de Y y no respecto de X ni Z, podemos determinar que el desplazamiento es vertical.
- Si varia tanto del eje X como del eje Y y no varia Z, podemos determinar que esta haciendo un movimiento circular.
- Por último, si no se desplaza en el eje X ni en el Y y si en el Z podemos determinar que es un desplazamiento en profundidad, siendo dicha profundidad obtenida a partir de los centroides más significativos seleccionados.

Posteriormente, se ha realizado un modelo que nos permita diferenciar e identificar el tipo de movimiento de la mano con los datos obtenidos a partir del seguimiento del ROI.

Para la realización de dicho modelo, se pueden clasificar de varias formas estos datos, como pueden ser mediante el algoritmo de clasificación de máquinas de vectores de soporte (SVM), los k vecinos más cercanos (KNN) o un sistema basado en reglas.

En nuestro caso, hemos decidido implementar el algoritmo de k - vecinos más próximos (**KNN**). Hemos elegido este modelo debido a que el algoritmo SVM actúa de forma binaria, es decir, que solo nos permite obtener los datos de si un elemento pertenece a un clase específica o no. Como en nuestro caso queremos que nos diferencie entre las cuatro clases de movimiento distintas, no resultaba factible la implementación de este algoritmo, cosa que el algoritmo KNN si nos permitía hacer.

Para poder realizar tanto el entrenamiento como el test, hemos necesitado realizar un dataset para cada una de estas fases destinando 8 secuencias para entrenar el modelo ("xtr" para las variables e "ytr" para las clases) y 4 para testearlo ("xt" para las variables e "yt" para las clases). En estos datasets, se han colocado 4 variables que describen cada columna: desplazamiento en el eje X, desplazamiento en el eje Y, diferencia de profundidad y las 4 clases correspondientes a cada movimiento.

Ejemplo de dataset utilizado:

	DespX	DespY	Depth	Clase
	...	...	...	1
	...	...	...	2
	...	...	...	3
	...	...	...	4

Para el algoritmo KNN, hemos utilizado un número de 3 vecinos mas cercanos (k = 3) medido por distancias euclídeas.

```
%% Algoritmo KNN - entrenamiento
modelo = fitcknn(xtr,ytr,'NumNeighbors',3,'Distance','euclidean');
%% Predicción del modelo
resultado = predict(modelo, xt);
%% Cálculo de aciertos
acierto = sum(resultado == yt)/length(yt)*100;
```

Los movimientos que nuestro modelo va a ser capaz de detectar son los siguientes:

- Desplazamiento frontal correspondiente a la clase 1.
- Desplazamiento circular correspondiente a la clase 2.
- Desplazamiento lateral derecha correspondiente a la clase 3.
- Desplazamiento hacia arriba correspondiente a la clase 4.

5. Ejecución y resultados

Para ejecutar el código simplemente necesitaremos tener los archivos de código: "clasificador.m", "createMaskColor.m", "createMaskDepth.m" y "Practical.m" y los datasets utilizados para el clasificador: "DStest.mat" y "DSentrenamiento.mat".

Para ejecutar, usaríamos el script "Practical.m" el cual llama a "clasificador.m" donde se crea nuestro clasificador. A continuación, se le aplica a una secuencia dada, concretamente en la linea 7, colocaremos la ruta de dicha secuencia y finalmente se muestra que movimiento ha sido detectado.

```
1  % DEI Practical assignment 2021/22
2  % Objective: 3D HCI Gesture learning
3
4  %% Acquisition
5  clasificador;
6  load modelo.mat;
7  load('secuencias/test/scan3d-ri-27Feb2014-094558.mat'); % Load dataset
```

Resultados obtenidos a partir de la secuencia dada:

Figura 6: Secuencia horizontal frame 2

Figura 7: Secuencia horizontal frame 26

```
>> Practical
Creando modelo...
Acierto del modelo = 100%
Secuencia cargada
Analizando secuencia...
Movimiento detectado:  Desplazamiento lateral derecha
>>
```

Figura 8: Ejecución trayectoria lateral derecha