ROBÓTICA Y PERCEPCIÓN COMPUTACIONAL

ENTREGA Final Análisis

Autores: Jason Felipe Vaz

Índice

1. Segmentación de la imagen

1.1. Etiquetado de imágenes

En primer lugar, se hace un etiquetado de imágenes para posteriormente poder entrenar al algoritmo. Para ello, utilizamos uno de los videos disponibles en moodle (video2017-3.avi) y lo paramos en el momento que consideramos para poder hacer un etiquetado de la imagen como se muestra en la siguiente captura:

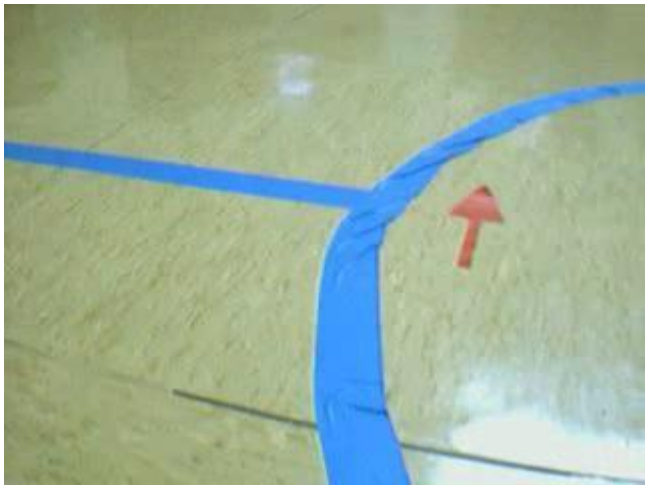

Imagen 1: Imagen Original video2017-3.avi

Imagen 2: Imagen Marcada video2017-3.avi

El procedimiento utilizado es muy sencillo, utilizamos las distintas funciones que nos da opencv, como son Cv2.VideoCapture para capturar el video que queremos utilizar para el marcado, Cv2.imshow para capturar la imagen que vamos a marcar y también utilizamos cv2.COLOR_BGR2RGB, para pasar el espacio de colores de BGR a RGB.

1.2. Distribución de colores en las imágenes

A continuación, se adjuntan las imágenes de las distribuciones.

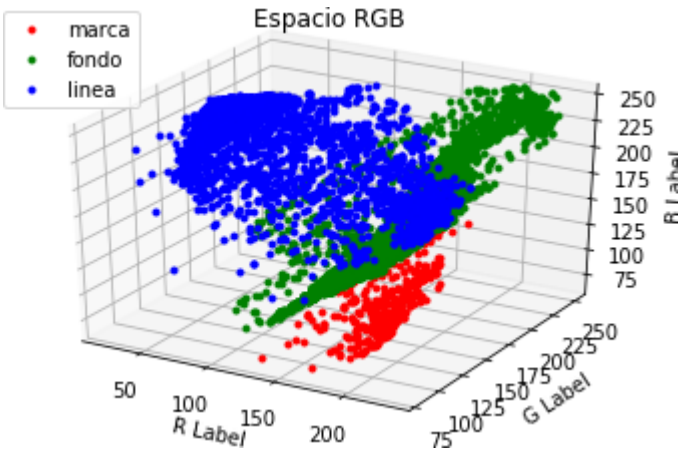

Imagen 3: Espacio RGB imagen entrenamiento video2017-3.avi

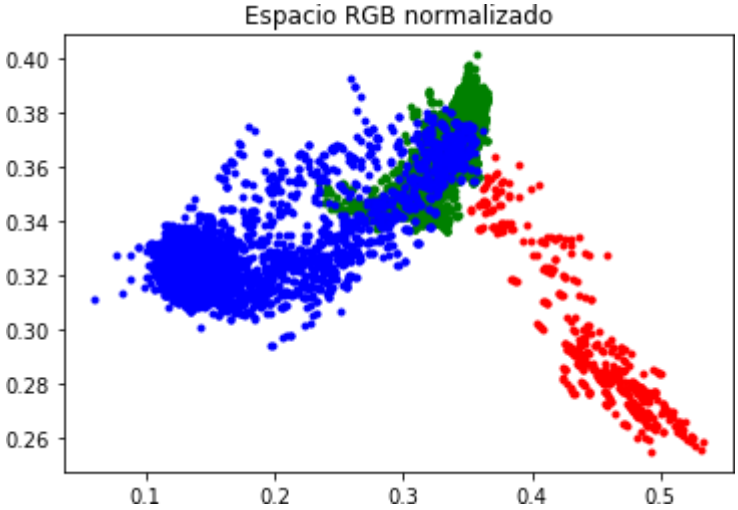

Imagen 4: Espacio Normalizado RGB imagen entrenamiento video2017-3.avi

Como podemos observar, aunque la distribución de los colores este superpuesta podemos observar que las nubes de puntos donde mayor concentración hay difieren unas de otras. Esta superposición también se debe a errores en la precisión de marcar con el ratón y también a un exceso de marcado.

Para poder pintar los datos, almacenamos los valores RGB fueron marcados en en rojo verde y azul, utilizando las siguientes líneas de código:

```
data_marca = imNp[np.where(np.all(np.equal(markImg,(255,0,0)),2))]
data_fondo = imNp[np.where(np.all(np.equal(markImg,(0,255,0)),2))]
data_linea = imNp[np.where(np.all(np.equal(markImg,(0,0,255)),2))]
```

Posteriormente se vuelve a hacer lo mismo, pero normalizando

1.3. Espacio de color

Como se ha mencionado en el apartado anterior el espacio de colores utilizado es el RGB, haciendo una conversión de colores utilizando la función de la biblioteca opencv cv2.COLOR_BGR2RGB.

1.4. Algoritmo de segmentación. Eficiencia

En primeras entregas de la práctica se hizo uno de la función inRange() pero esto provocaba que se eligieran píxeles en exceso, por lo que se descartó como opción, además de que en las sesiones de clase, se dejo aún más claro que usar esta función provocaría errores debido a que no se controla los píxeles exactos que vamos a tomar.

El algoritmo que hemos decidido utilizar ha sido uno de los recomendados en clase, el de la distancia Euclídea. Para ello se ha utilizado K-NeighborsClassifier de la biblioteca sklearn.

Para el entrenamiento se ha utilizado el video2017-3.avi utilizando como imágenes las mencionadas en el apartado 1.1.

En primer lugar, normalizamos la imagen marcada que hemos obtenido anteriormente utilizando la biblioteca NumPy y su función de rollaxis, para ello divido el valor RGB de cada píxel por la suma de los tres valores como se muestra en la siguiente línea de código, ya que rollaxis se centra en el uso de ejes para poder realizar su correspondiente traspuesta.

```
img_norm = np.rollaxis((np.rollaxis(orig_img, 2)+0.1)/(np.sum(orig_img, 2)+0.1), 0, 3
```

Almacenamos los valores de las imágenes normalizadas utilizadas y los concatenamos de la siguiente manera:

```
data_red = img_norm[np.where(np.all(np.equal(mark_img, (255, 0, 0)), 2))]
data_green = img_norm[np.where(np.all(np.equal(mark_img, (0, 255, 0)), 2))]
data_blue = img_norm[np.where(np.all(np.equal(mark_img, (0, 0, 255)), 2))]


data = np.concatenate([data_red, data_green, data_blue])

target = np.concatenate([np.zeros(len(data_red[:]), dtype=int),
                         np.ones(len(data_green[:]), dtype=int),
                         np.full(len(data_blue[:]), 2, dtype=int)])
```

Las dos variables anteriores las utilizamos para entrenar con NearestCentroid, además de que clasifican las muestras de los dígitos, es decir, para cada conjunto de datos corresponderá cierta muestra. En nuestro caso, para los dígitos:

- **De color rojo:** corresponderá una matriz de 0's.
- **De color verde**: corresponderá una matriz de 1's.
- **De color azul**: corresponderá una matriz de 2's.

Finalmente con los datos obtenidos se utilizará la función fit(), para entrenas con las muestras obtenidas.

```
clf = NearestCentroid()
clf.fit(data, target)
```

El siguiente paso es capturar el video y convertir los frames de BGR a RGB además también volvemos a hacer una nueva normalización de la imagen normalizada pero esta vez utilizando los frames anteriormente convertidos de BGR a RGB:

```
frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
img_norm = np.rollaxis((np.rollaxis(frame_rgb, 2)+0.0)/(np.sum(frame_rgb, 2)+0.0), 0, 3)
```

Por último, mostramos dos pantallas con el video segmentado y sin segmentar.

Se ha decidido utilizar este algoritmo ya que es uno de los que se mencionan en clase, además, desde el principio nos resultó ser el más fácil de utilizar obteniendo un resueltos muy aceptables para el objetivo de la práctica.
Comentario: Nos planteamos cambiar al clasificador, ya que estabas utilizando el clasificador euclídeo pero dicho clasificador no era tan fino, ya que provocaba algunos pequeños huecos. El clasificador que nos planteamos utilizar ha sido el clasificador Lineal, pero debido a la falta de tiempo no hemos desarrollado esta opción.

1.5. Video resultado

El video resultado se adjunta en el zip entregado con nombre "VideoSalida.mp4"

2. Análisis de imagen, clasificación de escenas y estimación de la consigna de control.

2.1. Suposiciones

Para la parte de consigna se han tomado las siguientes suposiciones:

- Se Supondrá que las líneas van a ser infinitas, de esta forma una línea solo podrá generar una única salida en el borde de la imagen.

- No existirán más marcas además de las ya proporcionadas, es decir, marcas que únicamente sean iguales a escaleras, mujer, hombre y flecha.

- Cuando existan cruces únicamente van a poder existir marcas del tipo flecha, que son las que indicarán la dirección. El resto de marcas no aparecerán en el cruce.

- Además, en nuestro caso, es importante destacar, que las marcas del tipo, van a depender del ángulo de la elipse que forme, ya que esto nos proporcionará información suficiente como para saber en qué dirección girar. Solución interesante que se proporcionó en la última sesión de clase.

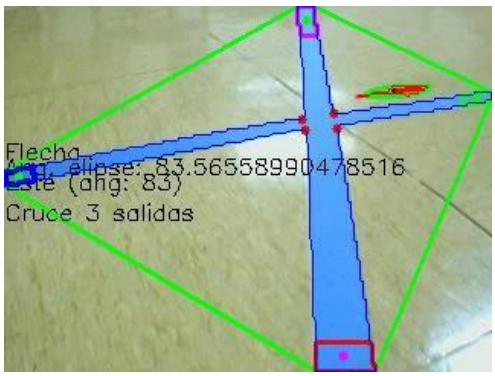

Imagen 5: intersecciones y flecha

En la **figura 1**, se muestran varias intersecciones, donde se pueden observar que se marcan en rojo las esquinas para su correspondiente análisis, y después se decide a qué lado girar, gracias a la información de la flecha (se explicará mejor el tema de la flecha posteriormente).
También, se puede observar en dicha imagen, que se forman distintas salidas las cuales se marcarán de distintos colores, esto se hace gracias a la detección de píxeles en el borde frame de la línea.

Continuando con las suposiciones:

- Hay que tener en cuenta, que las marcas (hombre, mujer, escalera, cruz y teléfono) se tratan de manera distinta que la flecha, ya que a las marcas se les pasará un clasificador, mientras que para la flecha se centrará en el uso de una elipse y de contornos para su correspondiente análisis**, por tanto, se tratará de manera distinta respecto al resto de marcas.**

- También como nos encontramos en la fase de consigna, damos por hecho que ya se ha realizado la fase de segmentación previamente, para poder usar los datos obtenidos en la fase de segmentación en esta fase de consigna.

2.2. Algoritmo de reconocimiento de la escena y los puntos de entrada y salidas de la misma

Los primeros pasos que se ha seguido, es encontrar las intersecciones que existen entre las líneas, y el borde de la imagen, de tal manera que podremos las salidas que existen, analizando las esquinas para poder determinar de cuantas líneas hay, si es solo una o varias, gracias a la información de las esquinas obtenidas.

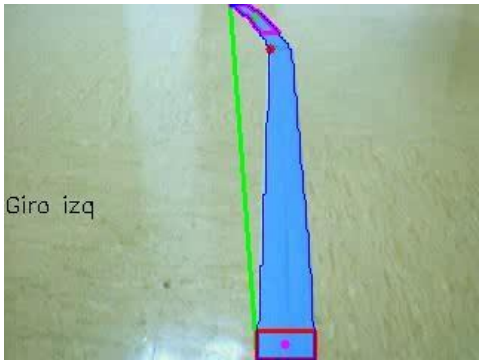

Imagen 6: Entrada y salida

En cuanto al algoritmos que se ha seguido para el reconocimiento de la escena simplemente nos hemos basado en el uso de la normalización previamente para a continuación usar la segmentación, con el objetivo de usar FindContours además del uso del ConveHll() para encontrar el casco convexo de la imágemes,por último basta con marcar las imagenés con el uso de DranContours().

Como se muestra a continuación, el contorno de la línea azul:

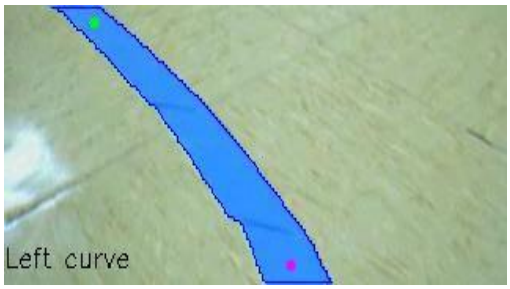

Imagen 7: línea y contorno

En cuanto, a las marcas (escalera, hombre…) ha sido también fácil ya que hay que realizar los mismos procedimientos ya que sabes los datos de sus correspondientes contornos y por tanto solo habría que marcarlas quedando de la siguiente manera:

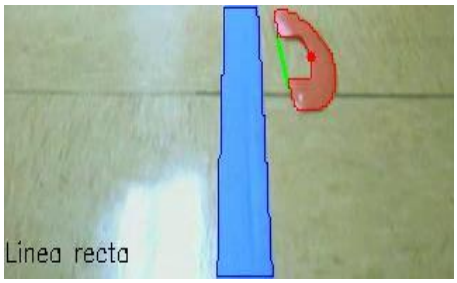

Imagen 8: teléfono

Imagen 9: persona

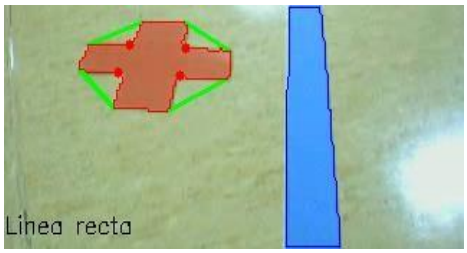

Imagen 10: Cruz

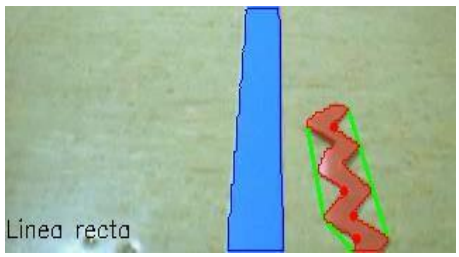

Imagen 11: Escalera.

En cuanto a los puntos de entradas y salidas, estarán determinadas por un vector que marcará la entradas y salidas durante el recorrido del robot. Esto se ha realizado partiendo la línea en 2 como en las transparencias de clase, para poder saber la mitad de la línea y finalmente marcar esos puntos clave, como de la siguiente manera:

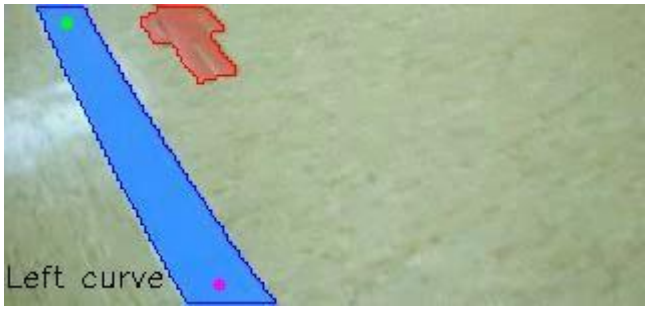

Imagen 12: Entrada y salida marcada

Las entradas como se pueden observar será marcadas de color rosa, mientras que las salidas existentes serán marcadas de color verde.

En cuanto a las curvas y rectas que se toman, cuando no existen intersecciones, se basarán en los bordes de las salidas, es decir, cuando la curva ocurra hacia la izquierda, por ejemplo, entonces tendrá una salida en el borde izquierda o superior de la imagen y la curva derecha ocurrirá lo mismo.

- Mientras que en las rectas tendrán salidas en el lado izquierda, superior o derecha.

2.3. Algoritmo de análisis de la orientación de la flecha y determinación de la salida elegida.

Para obtener la orientación de la flecha se obtendrá el contorno de la marca para identificarlo como flecha, una vez que se ha obtenido la marca de la flecha, se usará la función que realiza una elipse, de tal manera que podremos obtener parámetros necesarios para obtener el ángulo que definen la flecha y vectores que van desde el centro de la flecha hasta cada una de las salidas. Por tanto, el menor de estos ángulos va a corresponder con la salida a tomar.

Imagen 14: Análisis flecha e intersecciones

¿Pero qué ocurrirá en caso de que no exista flecha? Nosotros damos por hecho que siempre debería existir una flecha cuando haya cruces, es más, está dentro de nuestras suposiciones tomadas al principio de este documento.
Pero en caso de que no existan, tomará el camino con base a la distancia del centro superior de la imagen segmentada.

2.4. Estimación de la consigna de control

Como se puede observar en la siguiente imagen se utilizará vectores como consigna entre la entrada y la salida, la dirección de giro como ya se ha comentado estará determinada por la flecha. Luego la consigna que se usará será el ángulo que se forma entre la entrada y la salida para el giro del robot.

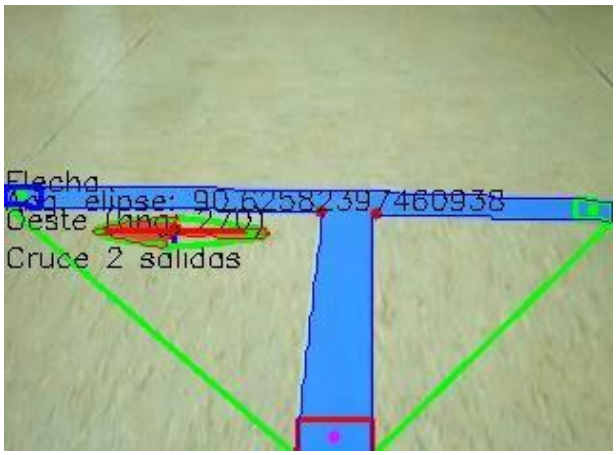

Imagen 13: Vector consigna

Comentario: la consigna que vamos a utilizar está basado en el ángulo de giro y las direcciones que vienen determinadas por las flechas, pero somos consciente que probablemente sea necesario el uso de una consigna donde involucre la velocidad, es decir, precisar una velocidad en función de la rotación obtenida en el robot. Si nos vemos capaces de implementarla queda pendiente para la entrega final

Destacar que, en el resultado del video, se especificará con un pequeño texto:

- El nombre de las marcas existente (flecha, cruz…)
- Grado de dirección de la flecha
- Giro izquierdo, derecho o recto
- Número de cruces existentes, sino hay no muestra nada
- La elipse

2.5. Vídeo resultado

El resultado de los videos tanto del análisis como de la consigna se encuentran en la carpeta denominada /Entrega2/VideosAnalisisConsginaMarca/Analisis.

3. Segmentación de un objeto circular y estimación de la distancia a la cámara

3.1. Algoritmo de segmentación y estimación del diámetro del círculo

Se han realizado los mismos procedimientos que se explican anteriormente en el apartado 1.4, cambiando una línea del código quedando como se muestra a continuación:

Línea anterior:

img_norm = np.rollaxis((np.rollaxis(frame_rgb, 2)+0.0)/(np.sum(frame_rgb, 2)+0.0), 0, 3)

Línea para el objeto circular:

img_norm = frame_rgb

Es decir, para la pelota no volvemos a normalizar una segunda vez.

Para este apartado se ha utilizado un video sacado de internet con un fondo homogéneo y una pelota.

La imagen marcada utilizada ha sido la siguiente:

Imagen 15: Imagen Original pelotaRodadora.avi

Imagen 16: Imagen Marcada pelotaRodadora.avi

Y su distribución:

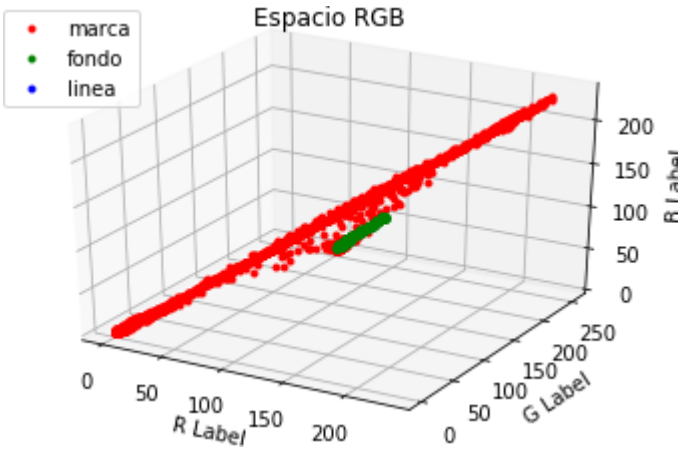

Imagen 17: Espacio RGB imagen entrenamiento pelotaRodadora.avi

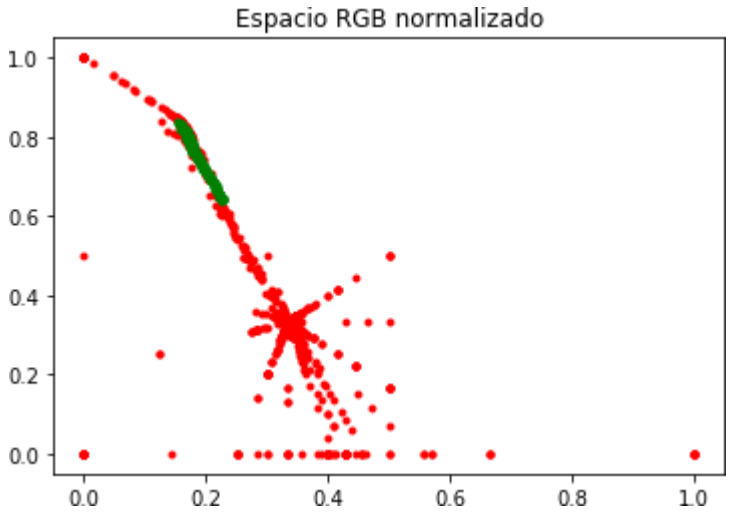

Imagen 18: Espacio RGB normalizado imagen entrenamiento pelotaRodadora.avi

Para el marcado y su distribución se usan los mismos métodos mencionados en el apartado 1.1 y 1.2.

Como podemos observar el espacio RGB sólo consta de dos colores rojo para la pelota y verde para el fondo. Al ser un ejemplo muy sencillo, se ve como las nubes de puntos de los distintos colores difieren unas de las otras.

3.2. Procedimiento de estimación de la distancia

Debido a la complejidad de utilizar el algoritmo con un objeto circular este apartado no ha sido realizado.

3.3. Vídeo demostrativo del resultado.

El video resultado se adjunta en el zip entregado con nombre “VideoSalidaPelota.mp4”

4. Reconocimiento de marcas

4.1. Procedimiento de descripción de la marca.

Primeramente, las marcas elegidas han sido las siguientes Cruz, Escalera, Personas y teléfono se ha usado la función huMoments() donde básicamente realiza el promedio medio de la imagen ponderado de las intensidades de los píxeles de la imagen ya que nuestras imágenes están pasadas a binarias.
Ya que esta función toma como entrada los momentos centrales de nuestra imagen.

4.2. Procedimiento de clasificación.

Como bien se ha explicado en el Punto 2, para la realización de las marcas se ha usado un clasificador de tipo euclídeo**.** Una vez aplicada la función huMoments() se realizan predicciones con los datos obtenidos a través de predict() como se muestra a continuación:

```
hu_mom = cv2.HuMoments(cv2.moments(newcnts_am[0])).flatten()
        pred = neigh_clf.predict([hu_mom])
```

Una vez obtenidos el análisis, y con la predicción previa solo tendremos que comprobar los valores obtenidos, según nuestra clasificación:

- 1 en la predicción corresponderá con **Escalera**.
- 2 en la predicción corresponderá con **Persona**.
- 3 en la predicción corresponderá con **Teléfono**.

Como ya se contó previamente aquí no se tendrá en cuenta la flecha ya que se trata de manera distinta.
Y por último se les pondrá un etiquetado de texto para distinguirlo en el video, todo esto se ha realizado de la siguiente manera:

```
elif pred == 1:
        cv2.putText(analy, "Escalera", (0, 100), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
(0, 0, 0), 1)
    elif pred == 2:
        cv2.putText(analy, "Persona", (0, 100), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
(0, 0, 0), 1)
    elif pred == 3:
        cv2.putText(analy, "Telefono", (0, 100), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
(0, 0, 0), 1)
```

Para la flecha como lo tratamos con la elipse y es un caso aparte se tratará de la siguiente manera:

```
else:
    cv2.putText(analy, "Flecha", (0, 100), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0,
0), 1)
```

Nota: En la carpeta chullFrames se podrán imágenes que contendrán el análisis de los distintos tipos de marcas, salida, entradas… Se han puesto solo imágenes que se veían necesarias ya que sino el fichero pesaba escesivamente.

Realmente en la práctica existen problemas con el hombre y la mujer, ya que no es capaz de diferenciarlos y los trata como iguales, provocando que en ocasiones los identifique como hombre y en otras ocasiones como mujer.

- Como por ahora no hemos podido solucionarlo, en esta entrega a las marcas de **hombre y mujer** se les va a identificar y englobar como **personas**, hasta que se encuentre alguna solución, como bien se muestra en la imagen 6.

También tenemos pensado cambiar a un clasificador del tipo LogisticRegression ya que en las sesiones de clase, a los compañeros que presentaban les funcionaba razonadamente mejor este tipo de clasificador, por tanto, quedará pendiente de implementar

4.3. Vídeo resultado entrenando conjunto de imágenes y evaluando en el vídeo que hay en el aula virtual.

El resultado de los videos tanto del análisis como de la consigna se encuentran en la carpeta denominada /consigna/Videos/analisisYconsigna.MP4
Dicho video contiene absolutamente todo el resultado correspondiente al análisis, marcas y consigna.

5. Conclusión final

Como conclusión final, para esta parte de segmentación, marcas, y consigna hemos llegado a distintas conclusiones de mejora como el cambio de clasificadores tanto en segmentación como en el reconocimiento de imágenes:

- En segmentación: se producen algunos huecos en el video, lo que hace que no sea del todo precisa la segmentación.

- En el reconocimiento de imágenes: no reconoce claramente a la marca del hombre y la mujer, por lo que causó que lo tuviéramos que tratar como personas al menos para esta entrega, por tanto, es algo a mejorar.

También existen otras posibles mejoras como ya se ha comentado en este documento, que es el tema de tener en cuenta la velocidad en la consigna en nuestra práctica.

Aunque como toda esta práctica realizada aún no llega a ser probado con el código del robot, no sabemos a ciencia cierta si funcionará por lo que se espera que tengamos que mejorar o cambiar algún algoritmo utilizado.

Centrándonos más en las dificultades obtenidas, hemos tenido problemas con las últimas entregas sobre todo con la parte de consigna, debido a que se nos hacía complejo en un principio saber lo que realmente había que "hacer" y sobre todo de que manera implementarla ya que es una práctica abierta, y se puede realizar de muchas maneras, aunque bien es cierto que las sesiones de clase nos sirvieron como apoyo a la hora de pensar como implementar ciertas cosas, como, por ejemplo, el tipo de clasificador a utilizar, como segmentar, suposiciones previas, como reconocer las marcas… Cabe destacar que en estas últimas entregas hemos sido influidos por la falta de tiempo como grupo.

En cuanto al trabajo grupal, somos conscientes de que la memoria es mejorable y se deberían definir y explicar en más profundidad dicha práctica, pero debido poco tiempo disponible en el grupo no hemos podido profundizar demasiado.