Universidad de Murcia

Facultad de Informática

— Visión Artificial —

Documentación Ejercicios

2016/2017

Autor
José María Sánchez Salas
josemaria.sanchez12@um.es

Profesor
Alberto Ruíz García
aruiz@um.es

Índice

1 Introducción

Este documento contiene la documentación de los ejercicios propuestos, así como de algunos expermientos realizados, de la asignatura Visión Artificial perteneciente a la mención de Computación del Grado de Ingeniería en Informática, impartido en la Facultad de Informática de la Universidad de Murcia en el curso académico 2016-2017.

La estructura de este documento es la siguiente: la primera sección contiene las explicaciones de los ejercicios propuestos, mientras que la segunda sección contiene las explicaciones de los experimentos que realizados, haciendo uso de lo aprendido en los ejercicios.

Es importante destacar, que **para todos los ejercicios**, para cerrar el programa hay que pulsar la tecla Escape (`Esc`), por eso mismo, en la resolución de los ejercicios, si es necesaria la pulsación de teclas, no se va a exponer el uso de esta tecla.

2 Ejercicios Propuestos

2.1 Ejercicio 1

Enunciado

Modificación de los canales de color (brillo, saturación, etc.) sobre la secuencia de imágenes tomada con la webcam.

Resolución

Este ejercicio ha sido resuelto de la siguiente manera: una vez capturada la imagen de la cámara, se hace una copia para evitar modificar la imagen obtenida, se cambia el espacio de color a HLS, se modifican los canales L y S, se convierte al espacio de color BGR y se muestra la imagen modificada.

2.2 Ejercicio 2

Enunciado

Amplía `player.py` para seleccionar con el ratón una región de interés (ROI) y almacenar en una lista imágenes, que pueden opcionalmente guardarse en disco. Este programa nos servirá como base para futuros ejercicios.

Resolución

Lo más importante de este ejercicio es cómo seleccionar una región de interés de la webcam. Para ello, hay que introducir un manejador de eventos de ratón; este manejador hace lo siguiente: cuando se pulsa el botón izquierdo, almacena las coordenadas del ratón como coordenadas iniciales del ROI, además se indica que se inicia el dibujo del ROI; mientras no se levante el botón, si se mueve el ratón, se van almacenando como coordenadas finales del ROI las coordenadas donde está el ratón, esto se hace así, para poder ir dibujando el ROI conforme se va creando; una vez que se levante el botón izquierdo, se almacena definitivamente las coordenadas finales y se añade, automáticamente dicho ROI a la lista de ROIs.

El programa principal, lo único que hace es: ver si se ha pulsado alguna tecla (y si es así, hacer la acción correspondiente); si se ha iniciado el dibujo del ROI, dibujarlo; mostrar la imagen de la cámara.

Las teclas que se procesan en este programa son:

- Tecla `Enter`: para eliminar el dibujo del ROI.
- Tecla `Espacio`: para parar la webcam en una imagen.
- Tecla `s`: para almacenar en disco los ROIs almacenados.

2.3 Ejercicio 3

Enunciado

Construye un clasificador de objetos en base a la similitud de los histogramas de color del ROI (de los 3 canales por separado).

Resolución

Haciendo uso del ejercicio anterior, la resolución de este ejercicio se basa simplemente en una vez seleccionado el ROI, se calcula y almacena su histograma, para comparar con el histograma del ROI de la imagen actual y si supera un determinado valor (variable global `maxDiff`), cambiar el color del ROI a azul, indicando que es similar al ROI almacenado.

Las teclas que se procesan en este programa son:

- Tecla `Enter`: para seleccionar el ROI y calcular su histograma.
- Tecla `Espacio`: para parar la webcam en una imagen.

2.4 Ejercicio 4

Enunciado

Construye un generador que detecte frames estáticos, o frames en movimiento, a partir de una secuencia de imágenes.

Resolución

Para la resolución de este ejercicio se ha hecho uso de la librería `umucv` proporcionada por el profesor, para crear un historial de las 9 últimas imágenes y mostrarlas de manera conjunta para que se vea el paso de los movimientos. Sin embargo, el cálculo para detectar movimiento, se hace únicamente con la imagen actual y la anterior. Una vez obtenidas la imagen actual y la anterior, se hace la diferencia entre ellas y si supera un determinado valor (variable global `maxDiff`), el programa muestra por terminal el mensaje `I caught you!`, indicando que ha habido movimiento.

2.5 Ejercicio 5

Enunciado

Construye un servidor web sencillo usando `flask` que muestre una cierta transformación de las imágenes tomadas con la cámara.

Resolución

Este ejercicio proporciona las siguientes transformaciones de imágenes:

- **Suavizado Gaussiano**. La dirección para realizar esta transformación es: `http://localhost:5000/transforms/gaussianBlur/<blurringFactor>` donde `blurringFactor` es el factor de suavizado con el que realizar la transformación. Este factor debe ser numérico, si no lo es, el servidor da un error.
- **Conversión a blanco y negro**. La dirección para realizar esta transformación es: `http://localhost:5000/transforms/convertToBW`.
- **Umbralización**. La dirección para realizar esta transformación es: `http://localhost:5000/transforms/threshold/<threshold>` donde `threshold` es el umbral con el que realizar la umbralización. Este umbral debe ser numérico, si no lo es, el servidor da un error.

– **Bordes**. La dirección para realizar esta transformación es:
`http://localhost:5000/transforms/edges/<thresholds>` donde `thresholds` son los umbrales necesarios para la transformación de bordes `Canny`. El formato de los umbrales es: `thresh1xthresh2`, donde `thresh1 < thresh2`. Ambos umbrales deben ser numéricos y cumplirse la condición anterior, si no es así, el servidor da un error.

2.6 Ejercicio 6

Enunciado

Implementar el efecto chroma con imágenes en vivo de la webcam. Pulsando una tecla se captura el fondo y los objetos que aparezcan se superponen en otra imagen o secuencia de video.

Resolución

La tecla para capturar el fondo es la tecla `Enter`. Tras pulsar esta tecla, el programa captura el fondo, crea el chroma y pone los objetos que aparezcan en la imagen `cube3.png`. Una característica interesante de este programa, es que proporciona un slider para ajustar el umbral con el que hacer el chroma.

Para hacer el chroma, tanto el fondo como la imagen actual se pasan al espacio de color YUV para eliminar la luminosidad y quedarnos con el color y la saturación, para así poder hacer un chroma más efectivo.

2.7 Ejercicio 7

Enunciado

Implementa la segmentación por color usando modelos de histograma en un programa que admite como argumento:

a) una carpeta con trozos de imagen que sirven como muestras de color y

b) otra imagen que deseamos clasificar.

El resultado puede ser un conjunto de máscaras para cada clase, o una "imagen de etiquetas", donde diferentes colores indican cada una de las regiones.

Resolución

La ejecución de este programa debe ser: `./exercise07.py [-h] models image`, donde:

– `[-h]` es una opción para mostrar la ayuda.

– `models` es la carpeta que contiene los modelos de color.

– `image` es la imagen que se quiere segmentar.

Tras ejecutar correctamente el programa, este cargará los modelos (como color del modelo, se obtiene el color medio de toda la imagen), leerá la imagen y la segmentará. Para segmentar la imagen, lo que hace es obtener las diferencias con todos los modelos, obtener para cada diferencia el modelo más cercano con él y segmentar la imagen con ese color. Una vez terminado, el programa mostrará la imagen original y la imagen segmentada en color. No se tratan todos los píxeles de la imagen, pues se filtran aquellos píxeles que no pertenecen a ningún modelo.

Los modelos para la ejecución de este ejercicio se encuentran en la carpeta `images/models`.

2.8 Ejercicio 8

Enunciado

Mostrar el efecto de diferentes filtros de imagen sobre la imagen en vivo de la webcam, seleccionando con el teclado el filtro deseado y permitiendo modificar sus posibles parámetros (p.ej. el nivel de suavizado) con las teclas de flecha.

Resolución

La realización de este ejercicio se ha hecho de la siguiente manera: puesto que todos los filtros deben de tener una tecla particular para aplicarse, todos tienen un valor inicial para poder realizar el filtro, un valor actual con el que se aplica el filtro, y para poder modificarse con las teclas de flechas, todos tienen sus valores de "paso" para modificar el valor actual del filtro; he decidido crear diccionarios con cada propiedad anterior: tecla del filtro, valor inicial, valor actual y valor de paso.

De esta manera, lo que consigo, es que en el código para la aplicación del filtro solamente trabaje con esos diccionarios y con una variable que indica el filtro a aplicar. Es decir, esté desacoplado al filtro real a aplicar. El programa principal, lo único de lo que se encarga es de procesar las teclas pulsadas para la aplicación del filtro correspondiente.

Es importante destacar que el programa hace división entre dos tipos de filtros: los que hacen uso de una matriz de convolución (suma, resta, multiplicación, división, bordes, suavizado horizontal y vertical, etc) y los que no (suavizado gaussiano) (de este tipo, solo está el suavizado gaussiano, pero está preparado para añadir más filtros). De esta manera, dependiendo del tipo de filtro que se haga, este se aplicará según sea conveniente.

Este programa se diferencia de los anteriores en que, una vez que se aplica un filtro, para salir de él hay que pulsar la tecla `Esc`. Esto hace que se deje de aplicar el filtro y aparecerá otra vez la imagen de la cámara original, listo para aplicar otro filtro. Para salir del programa, es necesario pulsar la tecla anterior cuando se encuentre la imagen de la camara original.

Las teclas para realizar los filtros son:

- Tecla `a`: para el filtro de sumar y restar.
- Tecla `b`: para el filtro de multiplicación y división.
- Tecla `c`: para el filtro de desplazamiento.
- Tecla `d`: para el filtro de bordes horizontales.
- Tecla `e`: para el filtro de bordes verticales.
- Tecla `f`: para el filtro de suavizado por media.
- Tecla `g`: para el filtro de suavizado gaussiano.
- Tecla `upArrow`: para incrementar el valor actual del filtro.
- Tecla `downArrow`: para decrementar el valor actual del filtro.
- Tecla `Esc`: para salir del filtro actual y para terminar el programa.

2.8.1 Extensión

La extensión del ejercicio 8 consiste en permitir aplicar los filtros a un ROI seleccionado y mostrar la imagen normal y el ROI con el filtro aplicado. De esta manera, se puede comparar el efecto de los filtros con la imagen original.

2.9 Ejercicio 9

Enunciado

Reconocimiento de objetos con la webcam basado en el número de coincidencias de keypoints. Pulsando una tecla se pueden ir guardando modelos (p. ej. carátulas de CD). Cuando detectamos que la imagen está bastante quieta, o cuando se pulse otra tecla, calculamos los puntos de interés de la imagen actual y sus descriptores y vemos si hay suficientes coincidencias con los de algún modelo.

Resolución

Este ejercicio proporciona la siguiente funcionalidad con el teclado:

- Tecla `s`: para almacenar el modelo que hay en el ROI seleccionado.
- Tecla `Enter`: para procesar la imagen actual en busca de los modelos seleccionados.

El programa, conforme se van seleccionando modelos, va mostrando el último modelo almacenado. Una vez que se quiera procesar la imagen actual, entonces, procesa la imagen en busca de los modelos, si alguno supera el mínimo de coincidencias (variable `minMatches` que se puede ajustar en el slider que aparece encima de la imagen principal), entonces se muestra en otra ventana la imagen del modelo con el que ha coincidido.

2.10 Ejercicio 10

Enunciado

Reconocimiento de formas con la webcam (o sobre un conjunto de imágenes) basado en descriptores frecuenciales.

Resolución

Este programa se debe ejecutar de la siguiente manera: `./exercise10.py [-h] [-d DEV] templates labels`, donde:

- `[-h]` es la opción de ayuda.
- `[-d DEV]` es la opción para indicar otro dispositivo de webcam (por defecto es el 0) o un directorio donde se encuentran las imágenes que se quieren procesar.
- `templates` es la imagen que contiene los modelos.
- `labels` son las etiquetas asociadas a cada modelo.

Este programa permite procesar tanto imágenes tomadas de la webcam como de un conjunto de imágenes (dentro de un directorio). Para el procesar las imágenes tomadas por la webcam, es necesario que se seleccione el ROI y se pulse la tecla `Enter` para procesarlo. En cambio, para la secuencia de imágenes, se procesa cada imagen completamente. El procesamiento de cada imagen se hace por separado, pulsando la tecla `n` se permite procesar la siguiente imagen.

En ambos casos, el programa, cuando procesa la imagen, muestra por consola el resultado obtenido. Una cosa importante a tener en cuenta en este ejercicio, es que como se pueden producir falsos positivos, estos se intercalan en la salida, por lo que suelen ensuciar el resultado obtenido.

2.11 Ejercicio 11

Enunciado

Rectifica la imagen de un plano para medir distancias (tomando referencias manualmente).

Resolución

Este ejercicio se ha planteado de la siguiente manera: teniendo en cuenta que no era muy lógico hacer un programa que solamente rectificara la imagen de las monedas `coins.png` o la de la falta `falta2.jpg`, ya que para cada caso concreto se necesita una imagen de referencia concreta para poder hacer la rectificación; se ha hecho de tal manera que el programa reciba por parámetros la imagen que se quiere rectificar y la imagen que va a usar como referencia. De esta manera, el programa es independiente de la imagen que se quiera rectificar y mostrará ambas imágenes para que el usuario pueda tomar los puntos de referencia de cada una.

Para la toma de los puntos de referencia, en todas las ventanas, se hace clickando con el ratón en la posición que se quiera. Una vez seleccionado el punto, este no se podrá mover ni eliminar, así que hay que tener cuidado a la hora de elegir los puntos. Sin embargo, el programa tiene la funcionalidad de eliminar todos los puntos escogidos (tecla `c` que se mencionará más abajo). También es importante saber que para que funcione, el número de puntos elegidos en la

imagen original y la de referencia debe ser el mismo (si no, el programa terminará con un mensaje de error, indicando el por qué); y la correspondencia entre dichos puntos es por orden de selección, es decir: el punto `p1` de la imagen original se corresponde con el punto `p1` de la imagen de referencia, y así con todos los puntos.

Una vez que se hayan seleccionado los puntos tanto en la imagen real como en la referencia, para realizar la transformación hay que pulsar la tecla `Enter`. Acto seguido aparecerá una ventana con la imagen transformada. Lo primero que hay que hacer, es seleccionar en dicha imagen, dos puntos de referencia (para que funcione correctamente, estos dos puntos deben pertenecer a la referencia en la imagen transformada) y tras esto, pulsar la tecla `d`. Después, el programa se quedará esperando a que se introduzca la distancia real que hay entre esos dos puntos seleccionados. Una vez introducidos, el programa pedirá que se seleccionen aquellos para los que nosotros queremos saber su distancia real, una vez seleccionados, hay que pulsar la tecla `d` para obtener la distancia real entre esos puntos.

Puesto que la transformación puede provocar que la imagen resultado salga en una posición que no se vea bien en la pantalla, el programa dispone de cuatro variables que se pueden modificar a gusto para modificar la posición y el tamaño de la imagen resultado en la ventana. Estas cuatro variables son:

- `dx, dy` para desplazar la imagen en el eje X e Y, respectivamente.
- `sx, sy` para escalar la imagen por un factor de escala en el eje X e Y, respectivamente.

Estas cuatro variables se tienen que modificar en el código del programa y se encuentran en la parte final de la declaración de todas las variables globales del programa, justo encima de la función `srcMouseEventHandler()`.

Para poder ejecutar correctamente el programa, este debe ejecutarse de la siguiente manera: `./exercise11.py [-h] source reference`, donde:

- `[-h]` es la opción de ayuda.
- `source` es la imagen que se quiere rectificar.
- `reference` es la imagen de referencia.

Las teclas necesarias para toda la funcionalidad de este programa son:

- Tecla `Enter`: para realizar la transformación (para ello, es necesario que se hayan seleccionado los mismos puntos en las dos imágenes, si no, el programa da un error).
- Tecla `d`: para indicar que se han seleccionado los puntos de referencia y para calcular la distancia entre puntos de la imagen transformada (para ello, es necesario que se hayan seleccionado al menos dos puntos en la imagen, si no, el programa da un error).
- Tecla `c`: para limpiar todos los puntos seleccionados en todas las imágenes.
- Usar el ratón para seleccionar los puntos.

2.12 Ejercicio 12

Enunciado

Crea automáticamente un mosaico panorámico ajustando varias imágenes (> 2). Recuerda que debe tratarse de una escena plana o de una escena cualquiera vista desde el mismo centro de proyección.

Resolución

Siguiendo la misma idea que en el ejercicio anterior, este ejercicio se ha realizado para que reciba como argumento del programa un directorio donde están almacenadas todas las imágenes con las que se quiera hacer la panorámica. Es importante destacar que para que la panorámica salga centrada, las imágenes deben seguir un orden concreto (por nombre): la primera imagen debe ser la que irá en el centro de la panorámica, las dos siguientes son aquellas fotos que van a derecha e izquierda (da igual el orden) de la central, las dos siguientes son aquellas que: una va a la derecha de la imagen derecha anterior y la otra va a la izquierda de la imagen izquierda anterior, y así sucesivamente. De esta

manera, el programa lo que va haciendo es construir la panorámica desde el centro hacia los lados.

Debido a que este programa hace uso del cálculo de puntos de interés para poder realizar las transformaciones adecuadas, hay que tener cuidado en que una imagen tenga solamente puntos de interés en común con la imagen a la que tiene que ir pegada, si no, el programa puede realizar un mosaico bastante extraño y fuera de ser una panorámica.

Independientemente del número de imágenes introducidas en la carpeta, el programa proporcionará una imagen de resolución `2000x600` píxeles. Aunque se puede modificar, ajustando las variables globables `sizeX, sizeY` que hay en el código del programa. Las imágenes con las que se ha realizado las pruebas de este programa se encuentran en la carpeta `images/panoramic`.

Para ejecutar este programa correctamente, se ha de ejecutar de la siguiente manera: `./exercise12.py [-h] folder`, donde:

- `[-h]` es la opción de ayuda.
- `folder` es la carpeta que contiene todas las imágenes con las que se quiere hacer la panorámica (respetando el orden anteriormente descrito, si se quiere que se haga una panorámica centrada).

2.13 Ejercicio 13

Enunciado

Crea un efecto de realidad aumentada dinámico.

Resolución

Para la resolución de este ejercicio, me he basado en el programa `pose0.py` proporcionado por el profesor. Este programa lo que hace es obtener de la webcam (o entrada de vídeo), las matrices de cámara asociadas al marcador que tiene la forma que muestra las imágenes de la Figura 1.

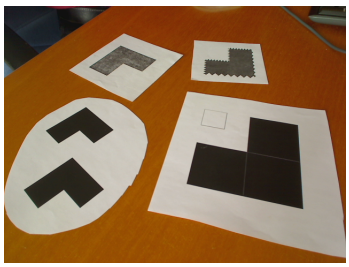

Figura 1: Imágenes del marcador para el ejercicio 13.

Basándome en esas matrices de cámara, lo que he realizado ha sido dibujar la imagen de la Figura 2 sobre cada uno de los marcadores encontrados, de manera que se haga un efecto de realidad aumentada (pues la imagen no está físicamente en esa posición). Este efecto de realidad aumentada se puede apreciar en la Figura 3.

Figura 2: Imagen dibujada en el ejercicio 13.

Para hacer el efecto de realidad aumentada dinámico, he implementado la posibilidad de que si selecciona un punto de la imagen, la imagen de la Figura 2 se desplace hasta esa posición. La parte más complicada de esto es cómo obtener el punto real de la escena a partir del punto que se ha seleccionado de la imagen.

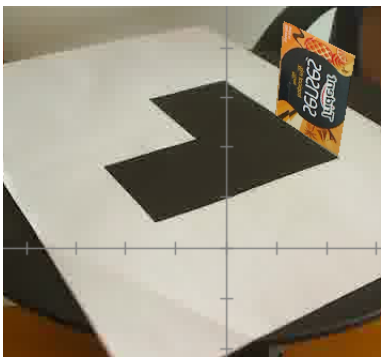

Figura 3: Efecto de realidad aumentada en el ejercicio 13.

Para ello, se ha hecho lo siguiente (todo este cálculo es para tranformar los puntos suponiendo que los puntos seleccionados de la imagen se encuentran en el mismo plano donde se sitúa el marcador). Dada la matriz de cámara:

$$M = \begin{bmatrix} x1 & y1 & z1 & w1 \\ x2 & y2 & z2 & w2 \\ x3 & y3 & z3 & w3 \end{bmatrix}$$

se ha obtenido la matriz de transformación (nos quedamos con las columnas 1,2 y 4):

$$H = \begin{bmatrix} x1 & y1 & w1 \\ x2 & y2 & w2 \\ x3 & y3 & w3 \end{bmatrix}$$

Con la matriz H, calculamos su matriz inversa H^{-1} que es la que transforma puntos de la imagen a puntos de la escena. Así pues, si se ha seleccinado el punto (p, q) de la imagen, el punto (x, y) se obtiene realizando la siguiente operación (aplicar la matriz de tranformación H^{-1}):

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = H^{-1} \cdot \begin{bmatrix} p \\ q \\ 1 \end{bmatrix}$$

que nos proporciona las coordenadas homogéneas del punto de la escena. Ahora solamente hace falta convertir esos puntos homogéneos a puntos normales, realizando la siguiente operación:

$$x = x/w \ ; \ y = y/w$$

Una vez obtenido el punto real de la escena, ahora simplemente hay que modificar las posiciones de la imagen virtual para que vayan hacia ese punto. Cuando se selecciona un punto, este punto aparece marcado en la escena para que se sepa a qué punto va la imagen.

Es importante destacar que el cálculo de este punto se hace para la primera matriz de cámara que se encuentra y luego ese punto se aplica a las demás cámaras. De tal manera que, en el caso de que se encuentren varios marcadores, las imágenes virtuales no se van a mover al mismo punto de la escena, sino que se van a mover al punto de su sistema de referencia, pues para cada marcador se obtiene una matriz de cámara que implica un sistema de referencia propio.

2.14 Ejercicio 14

Enunciado

Estima "a ojo" el campo de visión y parámetro f de tu cámara y comprueba que es consistente con el resultado de la calibración precisa mediante *chessboard*.

Resolución

La resolución de este ejercicio se ha hecho con la escena de la Figura 4. La botella se encuentra a una distancia de $142cm$ y su dimensión real es de $26cm$. En la imagen, la distancia que hay entre `p1` y `p2` es de $128px$. Teniendo estas medidas, la Figura 5 muestra el esquema de esta escena.

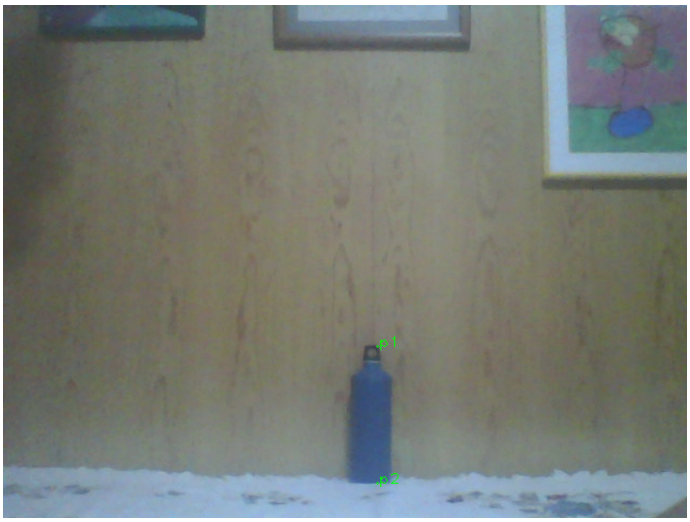

Figura 4: Escena para el ejercicio 14.

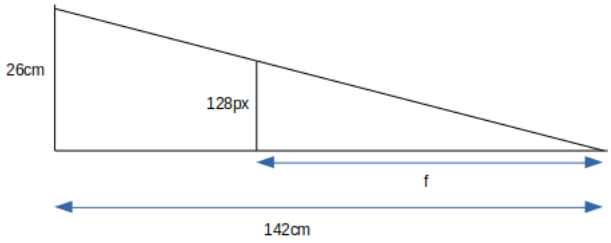

Figura 5: Escena para el ejercicio 14.

Ahora, lo importante es calcular f, que es el parámetro f de mi cámara. Para ello, hacemos uso de la igualdad triangular, teniendo que:

$$\frac{26}{142} = \frac{136}{f} \rightarrow f = \frac{136 \cdot 142}{26} = 742.7692 \approx 743$$

Teniendo en cuenta que la resolución de la cámara en la Figura 4 es de $640x480$, tenemos que la matriz de calibración de mi cámara es:

$$K = \begin{bmatrix} 743 & 0 & 320 \\ 0 & 743 & 240 \\ 0 & 0 & 1 \end{bmatrix}$$

También, tenemos que el campo de visión de mi cámara es:

$$\theta = 2 \arctan \frac{320}{743} = 0,8133544 \text{ radianes} \approx 46,6 \text{ grados}$$

3 Experimentos

Los experimentos que se han llevado a cabo se encuentran en la carpeta `experiments`.

3.1 Reconocedor de matrículas de coche

Enunciado

Programa de reconocimiento de matrículas con la webcam.

Resolución

Haciendo uso del ejercicio 10, se ha desarrollado este programa para reconocer matrículas de coche con la webcam. Para ello, primero se ha de seleccionar un ROI de la imagen, para después pulsar la tecla `Enter` y procesarlo. Hace uso de los modelos `platetemplates.jpg` que proporcionó el profesor y que se encuentra en la carpeta `images`

3.2 Emborronamiento de caras

Enunciado

Detección de caras con la webcam en vivo, con emborronamiento de las mismas.

Resolución

Para la resolución de este ejercicio se ha hecho uso de la herramienta de detección de caras `adaboost` (Enlace, última visita 24/04/2017). Como esta herramienta hay que cargarla como una librería, por defecto, se hace uso de la librería ubicada en la ruta: `$HOME/miniconda3/share/OpenCV/haarcascades/haarcascade_frontalface_default.xml`. Si se quiere utilizar la librería ubicada en otra ruta, el programa permite que se le pase como argumento la ruta usando la opción `-path` al ejecutarlo.

El funcionamiento del programa es simple: de cada frame obtenido por la webcam, detecta todas las caras y las emborrona usando un suavizado gaussiano. Sin embargo, también dispone de un modo en el que solamente se emborrona una única cara. Para esto, hay que pulsar la tecla `f`, seleccionar con un ROI la cara que se quiera emborronar y pulsar la tecla `Enter` para emborronar únicamente esa cara. Para poder realizar esto, se ha hecho uso del cálculo de los puntos de interés, de esta manera, para cada cara que se detecte, se calcula las coincidencias con la cara que se ha seleccionado. Si supera un mínimo de coincidencias (variable `minMatches` que se puede modificar en el slider de la parte superior de la ventana), entonces se emborrona únicamente esa cara.

La ejecución correcta de este programa es la siguiente: `./blurring-faces.py [-h] [-path PATH]`, donde:

- `[-h]` es la opción de ayuda.
- `[-path PATH]` es la opción para indicar la ruta donde se encuentra la librería de `adaboost`.

Las teclas necesarias para toda la funcionalidad de este programa son:

- Tecla `f`: para entrar en el modo de emborronamiento de una única cara.
- Tecla `Enter`: para que dentro del modo una cara y tras seleccionar la cara, se empiece a emborronar dicha cara.
- Tecla `Esc`: para salir del modo de una cara y salir del programa.
- Usar el ratón para seleccionar la cara.

4 Conclusiones

Debido a que la asignatura es plenamente práctica, la realización de todos los ejercicios, así como los experimientos, me han servido para afianzar los conocimientos de la asignatura. También a darme cuenta de que ciertos problemas que, a simple vista, son sencillos de resolver, cuando se intenta resolver mediante visión artificial, realmente no son tan sencillos. Y viceversa, problemas que cuando los ves piensas que son difíciles de resolver y luego se resuelven con diez simples líneas de código.

He de admitir que la asignatura me ha gustado bastante y sé que los conocimientos adquiridos me van a servir en un futuro, tanto en mi vida laboral como en mi vida personal, pues ya sé lo fácil que es hacer un programa que me detecte si hay movimiento en una habitación, donde tengo puesta una cámara grabando, y que el propio programa me envíe a mi móvil la imagen de lo que realmente está pasando en dicha habitación.

Bibliografía

[1] Alberto Ruíz García, *Apuntes de la asignatura.* Enlace (última visita 23/05/2017).

[2] OpenCV, *OpenCV Documentation.* Enlace (última visita 23/05/2017).

[3] Python, *The Python Tutorial.* Enlace (última visita 23/05/2017).