Proyecto Final: Laboratorio Visión por Ordenador I

Contenido

[1. Introducción 1](#_Toc186217850)

[2. Metodología 1](#_Toc186217851)

[3. Resultados 1](#_Toc186217852)

[4. Futuros desarrollos 1](#_Toc186217853)

# Introducción

A lo largo de la asignatura cuatrimestral hemos dado desde la calibración de una cámara, calculando sus parámetros intrínsecos y extrínsecos con Python hasta la detección de movimiento y el seguimiento de objetos.

En este proyecto final se desarrolla un proyecto muy completo que consiste en tres apartados: calibración, sistema de seguridad y sistema propuesto, todo a través de una *Raspberry Pi* y una *Camera module 3 WIDE (120º FOV).*

En el sistema de seguridad el objetivo consta de la detección correcta de un patrón a través de la extracción de las características, en esta parte se implementa un modulo que pueda detectar líneas, círculos, polígonos… Y por otra parte el descodificador que logre memorizar el patrón para determinar si se ha introducido la contraseña correcta, y ceder paso al bloque del sistema propuesto. En este último apartado se inicia el *tracker*, que consta en una simulación de una visión desde un coche, que detectará el movimiento de otros coches (de juguete, a los que la Raspberry Pi grabará e identificará) y los colores de los semáforos. (ESTO ESTÁ POR COMPROBAR)

# Metodología

Para la parte del sistema de seguridad se han creado tres módulos: *Figure.py*, *FigureDetector.py* y *Authenticator.py*, cada uno aprovechando funcionalidades de anterior. Con el módulo *Figure.py* se crean las figuras que el sistema va a ser capaz de detectar, en nuestro caso, un cuadrado verde, un círculo azul, un triángulo rojo y un pentágono morado. El módulo *FigureDetector.py* nos permite identificar individualmente cada una de las figuras anteriores, dependiendo de la figura que se introduzca en el detector. Como cada figura tiene un color diferente, su identificación se ha realizado en primer lugar mediante segmentación de color quedándonos con la máscara creada para posteriormente erosionar la imagen para eliminar posibles detecciones incorrectas y medir el número de esquinas detectadas con *cv2.findContours* y *cv2.approxPolyDP*. En función del número de esquinas detectadas se clasifica como un polígono u otro. Para el caso del círculo se calcula un coeficiente de circularidad y en función de si se encuentra dentro de unos límites, la figura detectada se clasifica como círculo o no. Además, para comprobar la correcta detección de cada figura se muestra sobre el frame la figura que se está detectando junto con su nombre (en el caso de detectar alguna figura). Para la parte de extracción de información, se ha diseñado el módulo *Authenticator.py*. Este módulo segmenta por los distintos colores de cada una de las figuras buscando alguna coincidencia. Si la hay (también en el número de esquinas), se almacena en una lista de patrones detectados, cuya coincidencia con la contraseña se verifica cuando hay 4 elementos en la lista. Si la secuencia de patrones detectados coincide con la contraseña, se muestra un rectángulo en verde alrededor del frame y se imprime por consola un mensaje indicando que se ha introducido la contraseña correcta durante 2 segundos. En caso contrario, se muestra un rectángulo rojo durante 2 segundos. Transcurridos esos 2 segundos el rectángulo se elimina y se puede volver a registrar otro patrón, hasta que se introduzca la contraseña correcta. Para evitar que se detecte el mismo patrón en una iteración del bucle y en la siguiente, se ha implementado un delay de 2 segundos desde la detección de un primer patrón, de forma que hasta que no transcurran esos 2 segundos el sistema no detectará ningún otro patrón.

# Resultados

# Futuros desarrollos