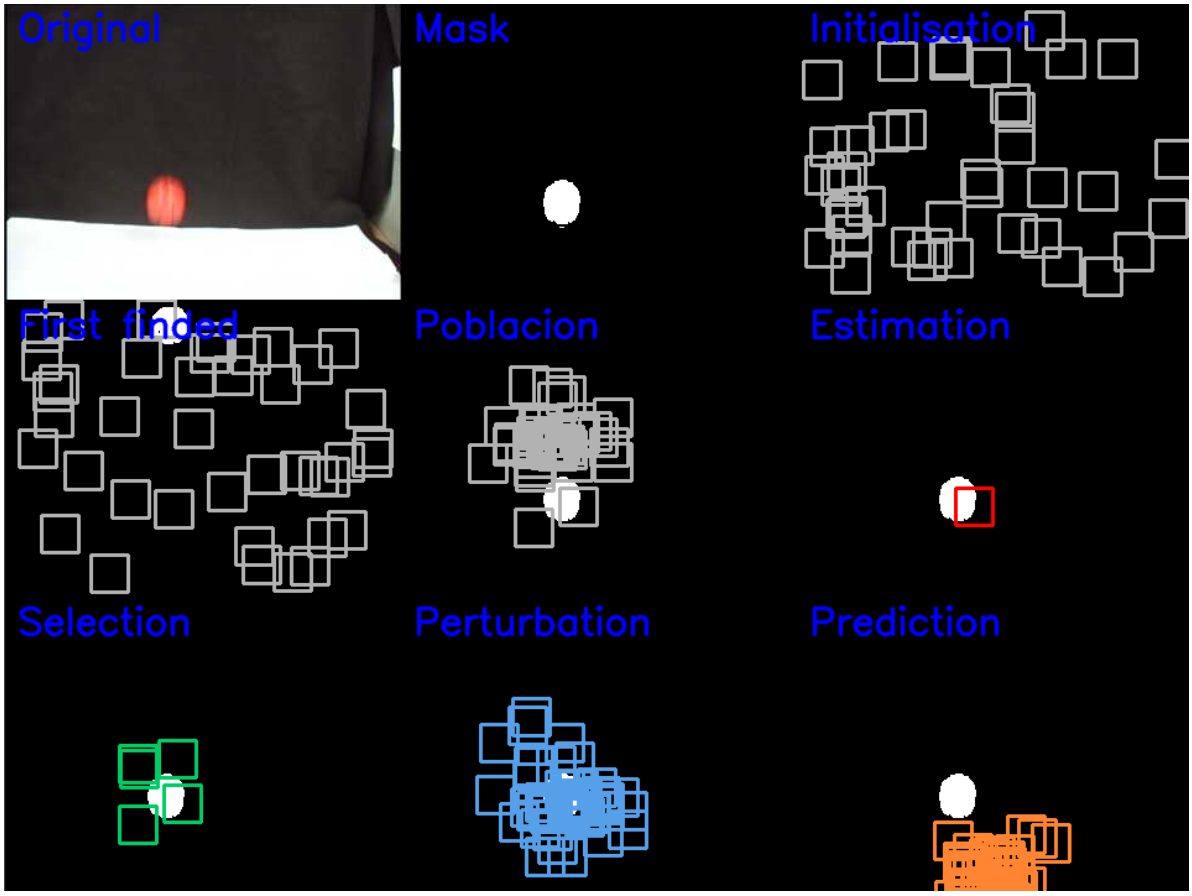
Práctica 2 Dinámica : Filtros de Partículas

Método de seguimiento visual



Luis ROSARIO TREMOULET

26/04/2022

URJC - Máster Universitario en Visión Artificial

Introducción

Este documento pretende resumir y explicar el desarrollo de un sistema de seguimiento visual basado en la metodología del filtro de partículas.

Para esta práctica he decidido utilizar el lenguaje Python (versión 3.9.2). Este proyecto se compone de tres archivos python:

- particle_class.py : Contiene la clase "Particle", que se crea para cada partícula generada

- filter_class.py : Contiene la clase principal "Filter", que gestiona todas las partículas en un momento t

- main_filter.py : Archivo principal que se utiliza para crear y gestionar las dos clases anteriores. También se encarga de mostrar el resultado final.

Para este proyecto solo se ha utilizado numpy (1.21.3) y opencv (4.5.5.24) como librerías externas.

Funcionamiento del algoritmo

Etapa 1 : Consideraciones iniciales + Filtro

Para empezar, he optado por utilizar la base de datos por defecto: la secuencia de imágenes de la pelota (disponible en mi proyecto o en el aula virtual)

Una vez pasada la ruta de la carpeta de la secuencia de imágenes como parámetro, el algoritmo lee todos los archivos .jpg de esa carpeta uno por uno.

Esas imágenes pasan una por una por el algoritmo principal.

Primero aplico una mascara (HSV) a la imagen original para poder detectar la pelota simplemente con un filtro de color.

Imagen 1 : Imagen original y imagen con filtro hsv

A partir de ahora, para entender mejor el funcionamiento de mi algoritmo, utilizaremos los mismos parámetros que son :

- Número de partículas = 40

- Tamaño de las partículas = 30x30 píxeles

- Tamaño de la perturbación = 20 píxeles

Etapa 2 : Inicialización

Una vez que tenemos el resultado del filtro de color, inicializamos N partículas (aquí 40) de tamaño WxH (aquí 30x30). Estas partículas tendrán una posición X e Y totalmente aleatoria dentro del tamaño de la imagen.

En este caso, las partículas están representadas por cajas delimitadoras.

Este paso continuará hasta que se encuentre la pelota o que el algoritmo la

pierda.

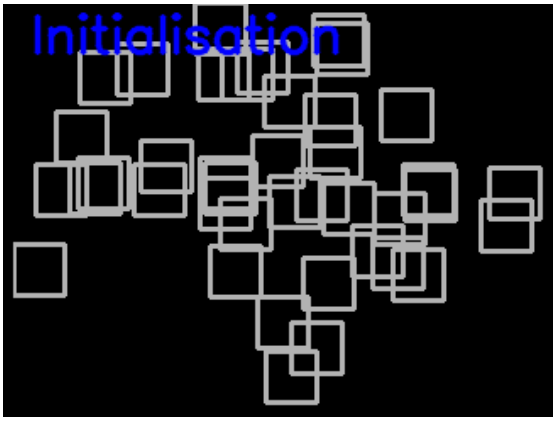


Imagen 2 : Representación de la inicialización

Etapa 3 : Evaluación

En este paso daremos un peso a todas las partículas. Aquí el peso se calculará en función del numero de píxeles blancos (del filtro de color) dentro de la partícula.

Así, una partícula que no toque la pelota tendrá un peso igual a 0 y una partícula que toque la pelota tendrá un peso igual al número de píxeles blancos que rodea

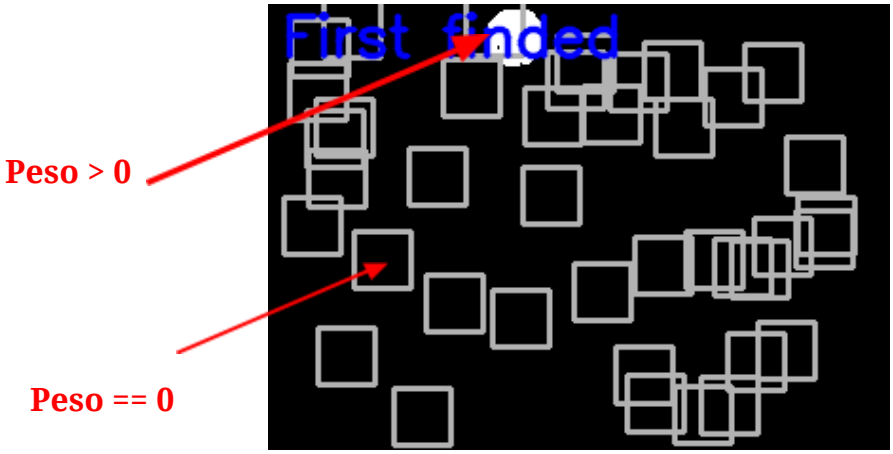


Imagen 3 : Pesos por particula

Etapa 4 : Estimación

Para este paso, simplemente seleccionamos la partícula con el mayor peso.

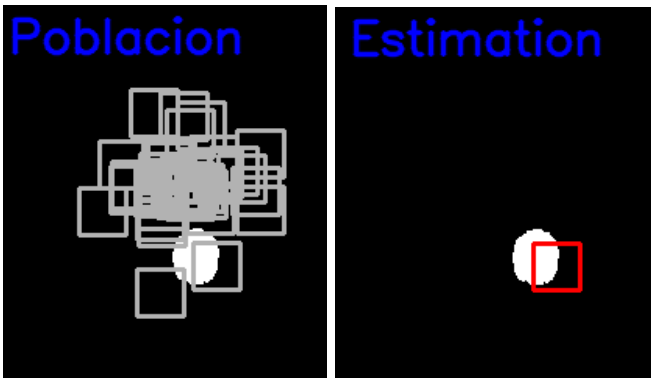


Imagen 4 : Representación de la estimación

Etapa 5 : Selección

En la etapa 5, seleccionamos las partículas con un peso superior a 0. He decido hacer el método de la ruleta en el paso siguiente (Para facilitar el desarrollo)

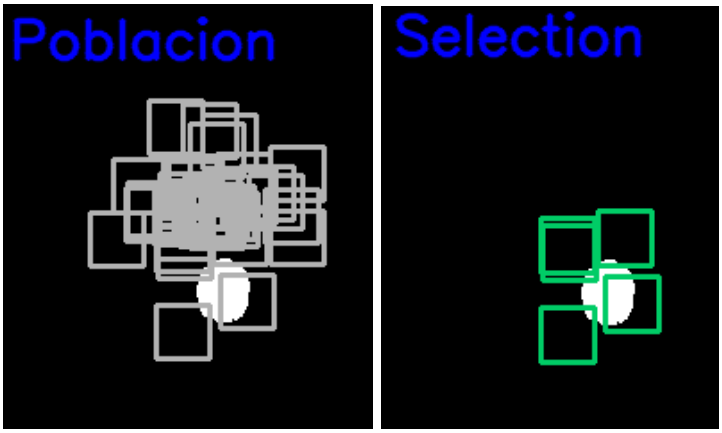


Imagen 5 : Representación de la selección

Etapa 6 : Difusión

Aquí es donde he decidido hacer el método de la ruleta. Utilizó el resultado del paso anterior : tomó todas las partículas con un peso mayor que 0. A continuación, elegiré al azar (con una proporcionalidad según el peso de cada partícula) N partículas (aquí 40). Estas partículas seleccionadas tendrán la misma posición que sus partículas "padre" (aquella de la que fueron seleccionadas). Luego aplico una perturbación aleatoria con una distribución gaussiana a todas estas nuevas partículas. El “tamaño” de la perturbación se puede elegir en los parámetros (Aquí 20 píxeles).

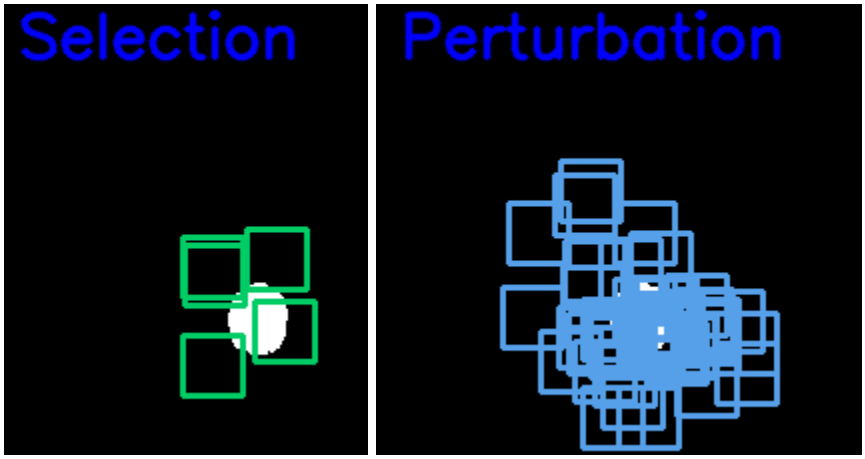


Imagen 6 : Representación de la perturbación

Las partículas del resultado de la perturbación se utilizarán como selección para la siguiente imagen.

Bonus : También he probado otros métodos de perturbación. Por ejemplo, en vez de utilizar una perturbación aleatoria con una distribución gaussiana de tamaño T, he intentado con un tamaño proporcional a las posiciones de las partículas con peso > 0.

```
x = int(np.random.normal(x, (x_max - x_min) // 2))
y = int(np.random.normal(y, (y_max - y_min) // 2))
```

x_max = coordenada x máxima de las partículas con peso > 0

y_max = coordenada y máxima de las partículas con peso > 0

x_min = coordenada x mínima de las partículas con peso > 0

y_min = coordenada y mínima de las partículas con peso > 0

Se puede probar añadiendo el parámetro "-p" antes de la ejecución del algoritmo (Más información en el Readme.md)

Etapa 7 : Predicción

Finalmente, para la predicción género N partículas (con una perturbación gaussiana) con una posición que depende de las partículas de las imágenes anteriores. Las coordenadas X y Y están calculadas en función de esas fórmulas :

X(t) = X(t-1) + (X(t-1) - X(t-2)) donde X y Y son la coordenadas

Y(t) = Y(t-1) + (Y(t-1) - Y(t-2)) y t el numero de la imagen

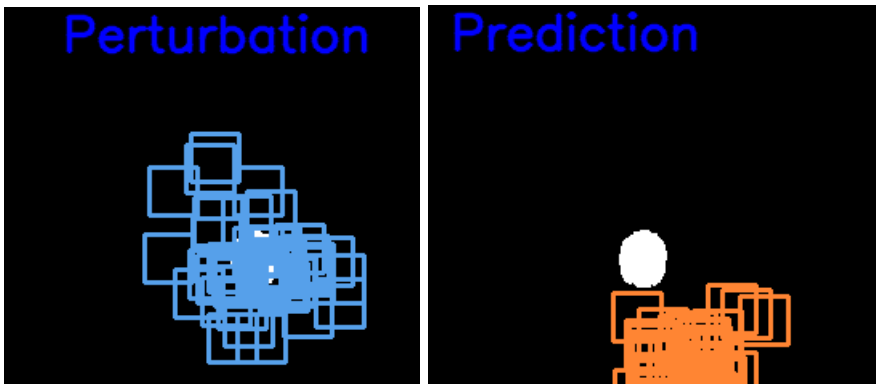


Imagen 6 : Representación de la predicción