```python
"""personne_detect.py | Robin Forestier | 8.03.2022

Detecting moving personne on video.
"""

# import OpenCV
import cv2


class PersonneDetect:
    """This class is used to detect people in a video stream."""
    def __init__(self):
        self.img = []
        self.copy = []
        self.detected = []
        self.backSub = cv2.createBackgroundSubtractorKNN(history=100, dist2Threshold=500.0, detectShadows=True)

    def img_to_gray(self):
        """If the image is in color, convert it to grayscale """

        if len(self.img.shape) == 3:
            self.img = cv2.cvtColor(self.img, cv2.COLOR_BGR2GRAY)
        else:
            pass

    def contour_detect(self, threshold):
        """Detect the biggest contours in the image and store them in a list

        :param threshold: The threshold image that was used
        :type threshold: numpy.ndarray
        """

        self.detected = []

        # Finding contours in the image.
        cnts, hierarchy = cv2.findContours(threshold, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

        # for eache contour
        for cnt in cnts:
            # if the perimeter is bigger than 100
            if 100 < cv2.arcLength(cnt, True) < 2000:
                # creting a bounding rect around it.
                # Creating a bounding rectangle around the contour.
                x, y, w, h = cv2.boundingRect(cnt)
                # store it
                self.detected.append([x, y, w, h])
                # draw a green rectangle.
                cv2.rectangle(self.copy, (x, y), (x + w, y + h), (0, 255, 0), 3)

    def personne_detect(self, img):
        """Detecting personne on image with background subtraction (KNN)

        :param img: The input image
        :type img: numpy.ndarray
        :return: the copy of the image with the green rectangle around the detected personne.
        :rtype: numpy.ndarray
        """

        self.img = img
        self.copy = img.copy()

        # Converting the image to grayscale if it is in color.
        self.img_to_gray()
        # Applying the background substractor to the image.
        fgmask = self.backSub.apply(self.img)
        # Blurring the image to remove the noise.
        blurImage = cv2.GaussianBlur(fgmask, (5, 5), 0)
        # Thresholding the image to make it binary.
        _, th = cv2.threshold(blurImage, 1, 255, cv2.THRESH_BINARY)

        # Realising 3 morphology transformation to clear the image of impure pixel.
        # To dilate the shape and close it.
        # kernel = np.ones((5, 5), np.uint8)
        kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (7, 7))
        th = cv2.morphologyEx(th, cv2.MORPH_OPEN, kernel)
        th = cv2.morphologyEx(th, cv2.MORPH_CLOSE, kernel)
        th = cv2.dilate(th, kernel, iterations=1)

        # call contour_detect for detect them.
        self.contour_detect(th)

        # return th copy of the img (with the green rectangle)
        return self.copy


if __name__ == '__main__':
    # Opening the video file.
    cap = cv2.VideoCapture("video_d.avi")
    # Creating an object of the class PersonneDetect.
    p = PersonneDetect()

    while True:
        # Reading the next frame from the video file.
        _, img = cap.read()
        # Resizing the image to a smaller size to make the algorithm faster.
        img = cv2.resize(img, (640, 480), interpolation=cv2.INTER_AREA)

        # Calling the function `personne_detect` of the class `PersonneDetect` and passing the image `img` as argument.
        img = p.personne_detect(img)

        # Showing the image in a window named "img".
        cv2.imshow("img", img)

        # Stop the program when the user press the key `q`.
        if cv2.waitKey(50) == ord("q"):
            break

    # Closing the video file and destroying all the windows.
    cv2.destroyAllWindows()
    cap.release()

```