```
# Noise filter threshold
# thresh = 1100
thresh = 1100

while(1):
    # read frames from both sources
    ret1, frame1 = cap1.read()
    ret2, frame2 = cap2.read()
    ret3, frame3 = cap3.read()

    dim = (480, 720)
    frame1 = cv2.resize(frame1, dim, interpolation=cv2.INTER_AREA)
    frame2 = cv2.resize(frame2, dim, interpolation=cv2.INTER_AREA)
    frame3 = cv2.resize(frame3, dim, interpolation=cv2.INTER_AREA)

    # Apply background subtraction
    fgmask_f1 = foog.apply(frame1)
    fgmask_f2 = foog.apply(frame2)
    fgmask_f3 = foog.apply(frame3)

    # Get rid of the shadows
    ret, fgmask_f1 = cv2.threshold(fgmask_f1, 250, 255, cv2.THRESH_BINARY)
    ret, fgmask_f2 = cv2.threshold(fgmask_f2, 250, 255, cv2.THRESH_BINARY)
    ret, fgmask_f3 = cv2.threshold(fgmask_f3, 250, 255, cv2.THRESH_BINARY)

    # Apply some morphological operations to make sure you have a good mask
    # fgmask = cv2.erode(fgmask,kernel,iterations = 1)
    fgmask_f1 = cv2.dilate(fgmask_f1, kernel, iterations=4)
    fgmask_f2 = cv2.dilate(fgmask_f2, kernel, iterations=4)
    fgmask_f3 = cv2.dilate(fgmask_f3, kernel, iterations=4)

    # Detect contours in the frame
    contours_f1, hierarchy_f1 = cv2.findContours(
        fgmask_f1, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    contours_f2, hierarchy_f2 = cv2.findContours(
        fgmask_f2, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    contours_f3, hierarchy_f3 = cv2.findContours(
        fgmask_f3, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    if contours_f1:

        # Get the maximum contour
        cnt = max(contours_f1, key=cv2.contourArea)
```