運動檢測技術（OpenCV 上的代碼）

原創 磐懟懟 深度學習與計算機視覺 2022-08-25 21:02 發表於河南

近年來，運動檢測技術已成為計算機視覺的重要研究領域之一。視頻序列上已經發明了許多方法，其中一些方法比其他方法更好。在本文中，我們將解釋並在OpenCV 上實現一些基本方法。

1. 幀差分

幀差分背後的想法非常簡單。我們逐像素檢查兩個視頻幀之間的差異。如果有運動，像素值就會發生變化，因此我們將獲得運動圖。

很簡單對嗎？但是，由於噪聲（例如照明的變化）可能會發生一些像素值變化，為了避免在我們的運動蒙版中捕獲噪聲，我們應用了一個閾值，該閾值基本上會突出強度方面的大變化並丟棄小的變化。請注意，閾值沒有正確的選擇，通常是憑經驗完成的。

現在我們理解了這個概念，讓我們展示一些代碼：

```
frames=[]
MAX_FRAMES = 1000
N = 2
THRESH = 60
ASSIGN_VALUE = 255 #Value to assign the pixel if the threshold is met

cap = cv2.VideoCapture(0)  #Capture using Computer's Webcam
```

```
for t in range(MAX_FRAMES):
    #Capture frame by frame
    ret, frame = cap.read()
    #Convert frame to grayscale
    frame_gray = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
    #Append to list of frames
    frames.append(frame_gray)
    if t >= N:
        #D(N) = || I(t) - I(t+N) || = || I(t-N) - I(t) ||
        diff = cv2.absdiff(frames[t-N], frames[t])
        #Mask Thresholding
        threshold_method = cv2.THRESH_BINARY
        ret, motion_mask = cv2.threshold(diff, THRESH, ASSIGN_VALUE, threshold
        #Display the Motion Mask
        cv2.imshow('Motion Mask', motion_mask)
```

這種方法具有計算性能，但是它存在兩個主要缺點：前景光圈和由幀速率和物體速度引起的重影。

Kameda 和Minoh 開發的一種解決方案是雙重差分，我們在時間t 和t-1 以及t-1 和t-2 之間的兩個幀之間操作閾值差異，然後將它們與邏輯AND 結合以確保我們始終檢測到一個對象而不是它的重影。

幀差分的另一個問題是，當對象停止移動時，它不再被檢測到。這顯然取決於我們想要完成的任務，但是假設我們想要繼續檢測移動物體，即使它停止了一段時間。這個問題的一個答案是背景減法技術。

2. 背景減法

背景減法是一種廣泛使用的方法，用於檢測靜態攝像機幀序列中的移動對象。它需要一個參考圖像來播放背景（通常在沒有對象的情況下獲取）。然後我們計算當前幀和背景幀（參考圖像）之間的差異。其主要任務是檢測通常代表運動物體的前景。

```
background = None
MAX_FRAMES = 1000
THRESH = 60
ASSIGN_VALUE = 255

cap = cv2.VideoCapture(0)
```

```
for t in range(MAX_FRAMES):
    # Capture frame-by-frame
    ret, frame = cap.read()
    # Convert frame to grayscale
    frame_gray = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
    if t == 0:
        # Train background with first frame
        background = frame_gray
    else:
        # Background subtraction
        diff = cv2.absdiff(background, frame_gray)
        # Mask thresholding
        ret, motion_mask = cv2.threshold(diff, THRESH, ASSIGN_VALUE, cv2.THRES
        # Display the motion mask and background
        cv2.imshow('Motion mask', motion_mask)
        cv2.imshow('Background', background)
```

背景減法代碼

如果对象的颜色与背景框不同，这种方法可以获得很好的效果。然而，像帧差分一样，它也有一些主要的缺点。毋庸置疑，它对光照变化和相机运动高度敏感，它还有一个“waking person"问题，这意味着如果背景物体（属于参考图像的物体）移动，则同时检测到真实物体及其重影.

在这种情况下，我们遇到了与帧差分相反的问题：“如果我们想停止检测前景对象并将其吸收到背景中怎么办？”

3. 自适应背景减法

这种方法基本上结合了之前的两种技术，通过引入学习率 λ 来充分利用两者。在每个时间步，我们对传入图像和先前背景的贡献进行加权以构建新背景。

例如，如果我们设置 λ=0.1，则在更新背景帧之前需要 10 帧（换句话说，前景对象将被吸收到背景中）。而对于 λ=0.5，我们有更快的更新（更新前只需要 2 帧）。请注意，选择 λ 没有规则，它是凭经验完成的，因为它取决于我们正在处理的任务和环境。

```
background = None
MAX_FRAMES = 1000
THRESH = 60
ASSIGN_VALUE = 255
```

```
ALPHA = 0.1

def update_background(current_frame, prev_bg, alpha):
    bg = alpha * current_frame + (1 - alpha) * prev_bg
    bg = np.uint8(bg)
    return bg

cap = cv2.VideoCapture(0)

for t in range(MAX_FRAMES):
    # Capture frame-by-frame
    ret, frame = cap.read()
    # Convert frame to grayscale
    frame_gray = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
    if t == 0:
        # Train background with first frame
        background = frame_gray
    else:
        # Background subtraction
        diff = cv2.absdiff(background, frame_gray)
        # Mask thresholding
        ret, motion_mask = cv2.threshold(diff, THRESH, ASSIGN_VALUE, cv2.THRES
        # Update background
        background = update_background(frame_gray, background, alpha = ALPHA)
        # Display the motion mask and background
        cv2.imshow('Motion mask', motion_mask)
        cv2.imshow('Background', background)
```

自适应背景减法代码

4. 高斯混合（MoG）

高斯混合是一种广泛使用的背景建模方法，用于从静态相机中检测运动物体。

简而言之，这种方法首先将每个像素建模为加权高斯的总和，其中权重定义了每个高斯的贡献。拥有多个高斯而不是一个的直觉是一个像素可以代表许多对象（例如雪花和后面的建筑物）。通过使用以前的帧计算颜色直方图，我们可以知道对象可能是背景或前景对象。

例如，当我们得到一个具有大权重和小标准偏差的高斯时，这意味着所描述的对象经常出现并且在帧之间没有变化，因此它可能是背景的一部分。这就是算法的工作原理；每个输入像素都会根据可

用模型进行检查。在匹配的情况下，我们更新模型的权重、均值和标准差，如果权重除以标准差很大，我们将像素分类为背景，否则分类为前景。

```
MAX_FRAMES = 1000
LEARNING_RATE = -1
fgbg = cv2.createBackgroundSubtractorMOG2()

cap = cv2.VideoCapture(0)

for t in range(MAX_FRAMES):
    # Capture frame-by-frame
    ret, frame = cap.read()
    #Apply MOG
    motion_mask = fgbg.apply(frame, LEARNING_RATE)
    #Get background
    background = fgbg.getBackgroundImage()
    # Display the motion mask and background
    cv2.imshow('background', background)
    cv2.imshow('Motion Mask', motion_mask)
```

混合高斯码

Github 存储库：https://github.com/safaabbes/Motion-Detection-Techniques-using-OpenCV

☆ END ☆

如果看到這裡，說明你喜歡這篇文章，請轉發、點贊。微信搜索「uncle_pn」，歡迎添加小編微信「woshicver」，每日朋友圈更新一篇高質量博文。

↓掃描二維碼添加小編↓

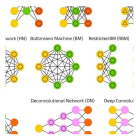
周烨

扫一扫上面的二维码图案，加我微信

喜歡此內容的人還喜歡

收藏| 機器學習最全知識點匯總（萬字長文）

小白學視覺



BeiT v2 來襲| BeiT升級，全面超越MAE，實現Vision Transformer 微調自由！

集智書僮

Matplotlib 可視化必備神書，完整PDF下載

小白學視覺