

Video Processing

Dr. Mongkol Ekpanyapong

Background subtraction

- The idea is to subtract between known foreground and the moving object
- The challenge is the constant changing of the light for outdoor environments
- Gaussian-Mixture based background model is introduced by P. KadewTraKupong in 2001



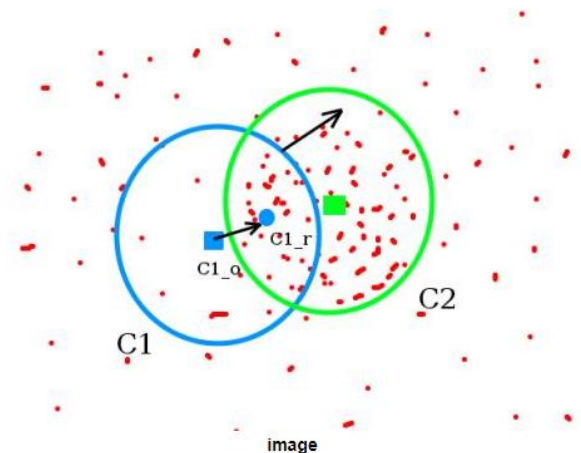
A blue square and a circuit board.

```
import numpy as np
import cv2
cap = cv2.VideoCapture('vdo/vtest.avi')
fgbg = cv2.createBackgroundSubtractorMOG2()
while(1):
    ret, frame = cap.read()
    fgmask = fgbg.apply(frame)
    cv2.imshow('frame',fgmask)
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break
cap.release()
cv2.destroyAllWindows()
```



Meanshift

- The idea of the meanshift is to calculate a pixel distribution (e.g., histogram)
- From a small window (or a circle), you have to move pixel window in the area of maximum pixel density
- If it doesn't match, we move it again, and continue the iterations such that center of window and its centroid falls on the same location



A blue square and a circuit board.

```
import numpy as np
import cv2
cap = cv2.VideoCapture('vdo/slow1.mp4')
# take first frame of the video
ret,frame = cap.read()
# setup initial location of window
r,h,c,w = 250,90,200,125 # simply hardcoded the values
track_window = (c,r,w,h)
# set up the ROI for tracking
roi = frame[r:r+h, c:c+w]
hsv_roi = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)
mask = cv2.inRange(hsv_roi, np.array((0., 60.,32.)), np.array((180.,255.,255.)))
roi_hist = cv2.calcHist([hsv_roi],[0],mask,[180],[0,180])
cv2.normalize(roi_hist,roi_hist,0,255,cv2.NORM_MINMAX)
# Setup the termination criteria, either 10 iteration or move by atleast 1 pt
term_crit = ( cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 1 )
```

Three small images: a circuit board, a yellow square, and a circuit board.

A small image in the top-left corner showing a blue sky and a red, textured surface.

```
while(1):
    ret ,frame = cap.read()
    if ret == True:
        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
        dst = cv2.calcBackProject([hsv],[0],roi_hist,[0,180],1)
        # apply meanshift to get the new location
        ret, track_window = cv2.meanShift(dst, track_window, term_crit)
        # Draw it on image
        x,y,w,h = track_window
        img2 = cv2.rectangle(frame, (x,y), (x+w,y+h), 255,2)
        cv2.imshow('img2',img2)
        k = cv2.waitKey(60) & 0xff
        if k == 27:
            break
    else:
        break
cv2.destroyAllWindows()
cap.release()
```



Camshift

- The problem of Meanshift is that the window is always fixed size. When object is moving closer, it should be adjusted
- Continuously Adaptive Meanshift) published by Gary Bradsky in 1988



A blue square and a circuit board.

```
import numpy as np
import cv2
cap = cv2.VideoCapture('vdo2/slow1.mp4')
# take first frame of the video
ret,frame = cap.read()
# setup initial location of window
r,h,c,w = 250,90,300,125 # simply hardcoded the values
track_window = (c,r,w,h)
# set up the ROI for tracking
roi = frame[r:r+h, c:c+w]
hsv_roi = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)
mask = cv2.inRange(hsv_roi, np.array((0., 60.,32.)), np.array((180.,255.,255.)))
roi_hist = cv2.calcHist([hsv_roi],[0],mask,[180],[0,180])
cv2.normalize(roi_hist,roi_hist,0,255,cv2.NORM_MINMAX)
# Setup the termination criteria, either 10 iteration or move by atleast 1 pt
term_crit = ( cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 1 )
```




```
while(1):
    ret ,frame = cap.read()
    if ret == True:
        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
        dst = cv2.calcBackProject([hsv],[0],roi_hist,[0,180],1)
        # apply meanshift to get the new location
        ret, track_window = cv2.CamShift(dst, track_window, term_crit)
        # Draw it on image
        pts = cv2.boxPoints(ret)
        pts = np.int0(pts)
        img2 = cv2.polylines(frame,[pts],True, 255,2)
        cv2.imshow('img2',frame)
        k = cv2.waitKey(60) & 0xff
        if k == 27:
            break
    else:
        break
cv2.destroyAllWindows()
cap.release()
```



Optical Flow

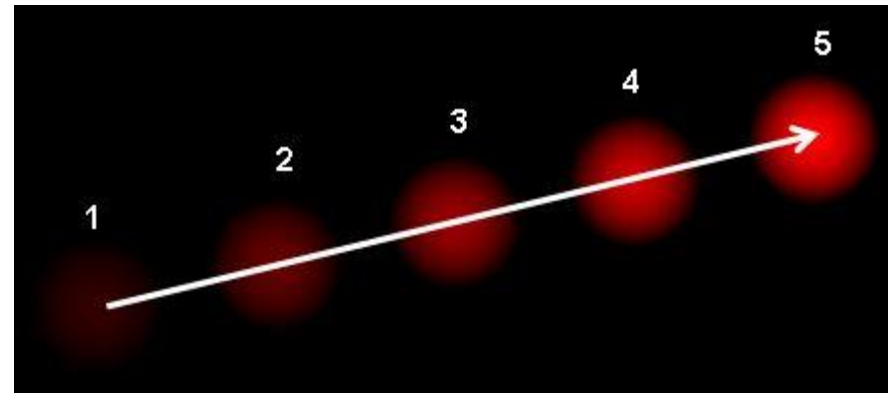
- Optical flow is the pattern of motion of image objects between consecutive frames
- It is 2D vector fields where each vector is a displacement vector showing the movement of points

$$I(x, y, t) = I(x + dx, y + dy, t + dt)$$

$$f_x u + f_y v + f_t = 0$$

$$f_x = \frac{\partial f}{\partial x} ; f_y = \frac{\partial f}{\partial y}$$

$$u = \frac{dx}{dt} ; v = \frac{dy}{dt}$$





Lucas-Kanade Optical Flow

- We use pyramids so that small motions are removed
- Hence, we get optical flow along with the scale



```
import numpy as np
import cv2
cap = cv2.VideoCapture('vdo/slow1.mp4')
# params for ShiTomasi corner detection
feature_params = dict( maxCorners = 100,
                        qualityLevel = 0.3,
                        minDistance = 7,
                        blockSize = 7 )
# Parameters for lucas kanade optical flow
lk_params = dict( winSize  = (15,15),
                  maxLevel = 2,
                  criteria  = (cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10,
0.03))
# Create some random colors
color = np.random.randint(0,255,(100,3))
# Take first frame and find corners in it
ret, old_frame = cap.read()
old_gray = cv2.cvtColor(old_frame, cv2.COLOR_BGR2GRAY)
p0 = cv2.goodFeaturesToTrack(old_gray, mask = None, **feature_params)
# Create a mask image for drawing purposes
mask = np.zeros_like(old_frame)
```



```
while(1):
    ret,frame = cap.read()
    frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # calculate optical flow
    p1, st, err = cv2.calcOpticalFlowPyrLK(old_gray, frame_gray, p0, None, **lk_params)
    good_new = p1[st==1]
    good_old = p0[st==1]
    for i,(new,old) in enumerate(zip(good_new,good_old)):
        a,b = new.ravel()
        c,d = old.ravel()
        mask = cv2.line(mask, (a,b),(c,d), color[i].tolist(), 2)
        frame = cv2.circle(frame,(a,b),5,color[i].tolist(),-1)
    img = cv2.add(frame,mask)
    cv2.imshow('frame',img)
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break
    # Now update the previous frame and previous points
    old_gray = frame_gray.copy()
    p0 = good_new.reshape(-1,1,2)
cv2.destroyAllWindows()
cap.release()
```





Questions?

