

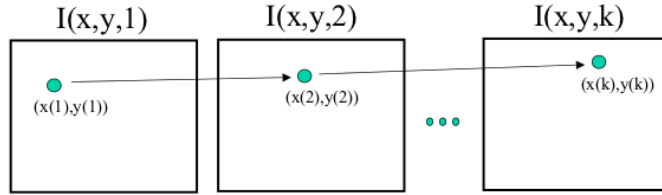
Piksel Takibi, Optik Akis, Lucas Kanade Algoritmasi

Hareket halindeki bir kameranın aldigi görüntülerdeki herhangi bir pikseli nasıl takip ederiz?

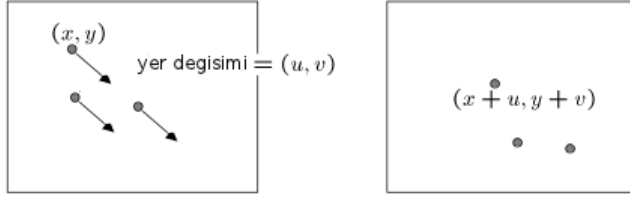
Matematiksel olarak temsil etmek gerekirse, zamana göre değişen 2 boyutlu görüntüyü bir fonksiyon olarak düşünelim, ki bu fonksiyonun değerleri ayrık-sal olarak, imajın ta kendisi. Bir $I(x(t), y(t), t)$ fonksiyonu piksel değerlerini veriyor. Bu fonksiyonda x, y ekran koordinatlarına tekabül ediyor, t ise zaman, 1, 2, .. gibi değerleri indeks değerleri var, mesela $I(100, 200, 1)$, bize 1. video karesindeki $x = 100, y = 200$ koordinatlarındaki piksel değerini verecek.

x, y değişkenleri parametrize edildi, bir noktayı takip etmek istiyoruz çünkü, ve t 'ye göre bu takip edilen noktanın x, y koordinatları belli bir gidisat yönünde değişiyor.

Su faraziye yapılarak takip problemimizi kolaylaştırabiliriz. Diyelim ki takip edilen bir nokta, görüldüğü her karede aynı piksel rengindedir. Bu çok sıradisi bir faraziye değil, resim karelerinden bir araba geçiyor mesela, ve bu arabanın üzerindeki piksellerin renkleri, en azından iki kare arasında değişmiyor. Işık seviyesi, golgede olma, vs. gibi durumlarda biraz değişebilir, fakat basitleştirme amacıyla bu faraziye geçerlidir.



Bir diğer faraziye, kameralar hareket ettiklerinde alınan iki görüntü arasındaki tüm piksellerin yer değişimi genellikle aynı yönde olmasıdır. Bu değişim yönünü $\langle u, v \rangle$ vektörü olarak görebiliriz, ve bu değişkenler iki görüntü arasındaki değişimde tüm pikseller için aynı olacaktır. Bu da normal, kamerayı alıp mesela sağa doğru hareket ettiriyoruz, ve görüntüdeki tüm pikseller sola doğru gidiyorlar.



Tüm bunları modelimizde nasıl kullanırız?

Takip edilen nokta her karede aynı renkte ise, şu ifade doğru demektir

$$I(x(t), y(t), t) = \text{sabit}$$

Eğer bu fonksiyonun zamana göre türevini alırsak

$$\frac{d I(x(t), y(t), t)}{dt} = 0$$

sonucu gelir. Esitliğin sağı sıfır, çünkü bir sabitin türevini aldık. Sol tarafa Zincirleme Kanununu uygularsak,

$$\frac{\partial I}{\partial x} \frac{dx}{dt} + \frac{\partial I}{\partial y} \frac{dy}{dt} + \frac{\partial I}{\partial t} = 0$$

Bu formülde dx/dt ve dy/dt , hareket halindeki (zaman geçerken) noktanın sonsuz küçüklikteki yer değişimi. Ayriksal bağlamda arka arkaya iki kare içindeki yer değişimi. O zaman,

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

Alttakiler ise mesafesel (spatial) gradyanlardır, bunların nasıl hesaplanacağını çok iyi biliyoruz!

$$\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}$$

Alttaki ise resim karelerinin zamana göre türevidir.

$$\frac{\partial I}{\partial t}$$

Daha derli toplu olarak göstermek gerekirse ana formül nihai olarak şöyle

$$I_x u + I_y v + I_t = 0$$

ya da

$$\nabla I \cdot \langle u, v \rangle = -I_t$$

Simdi u, v 'nin hesaplanmasina gelelim. Ustteki formulu bir veri noktası için yazmak yeterli değil. Ama bu formulu hem takip ettigimiz, hem de onun etrafındaki pikseller için yazarsak (onların yer değişimi de aynı değil mi?), ve bu sistemi cozersek, sonuca varabiliriz.

İki tane bilinmeyenimiz var, ama böylece pek çok formül elde ediyoruz. Veriler gürültülü olduğu için, aslında bilinmeyenden “daha fazla” formül elde etmek iyi, bu tür denklem sistemlerine “çok esitlige sahip (overdetermined)” denir, ve böyle tür sistemler En Az Kareler (Least Squares) ile çözülür. Tüm bunları bir araya koyunca su ortaya çıkar.

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_1) \\ \vdots & \vdots \\ I_x(p_k) & I_y(p_k) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_k) \end{bmatrix}$$

Gradyanların belli noktalarda hesaplandığını unutmayalım, o sebeple p_1, p_2 gibi piksel noktalarını bu fonksiyonlara geçiriyoruz.

Bu sistemi

$$A d = b$$

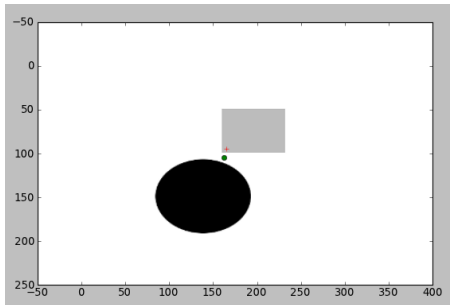
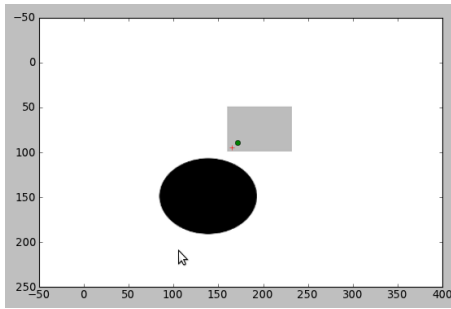
olarak gösterebiliriz, ki $d = \langle u, v \rangle$. Sol tarafı A^T ile çarpalım

$$A^T A d = A^T b$$

Eğer $A^T A$ 'nin matris tersini iki tarafla çarparsak, d yalnız kalır, ve sonuç elde edilir.

Bu denklemi Python Numpy'da `pinv` kullanarak cozeriz.

Test için üç tane resim kullandık, bu resimlerden `flow1-bw-0.png` başlangıç resmi, bu resmin ortasındaki objeleri GIMP kullanarak elle kopyaladık, bir üst sağ capraza doğru, bir alt sol capraza doğru, ve iki yeni resim elde ettik (`upright.png`, `dleft.png`). Takip edilen nokta gri dortgenin alt sol köşesinde. Lucas Kanade algoritması bu noktayı takip ederek, yeşil ile işaretledi.



```

import numpy as np
import scipy.signal as si
from PIL import Image

def gauss_kern():
    h1 = 15
    h2 = 15
    x, y = np.mgrid[0:h2, 0:h1]
    x = x-h2/2
    y = y-h1/2
    sigma = 1.5
    g = np.exp( -( x**2 + y**2 ) / (2*sigma**2) );
    return g / g.sum()

def deriv(im1, im2):
    g = gauss_kern()
    Img_smooth = si.convolve(im1,g,mode='same')
    fx,fy=np.gradient(Img_smooth)
    ft = si.convolve2d(im1, 0.25 * np.ones((2,2))) + \
        si.convolve2d(im2, -0.25 * np.ones((2,2)))

```

```

    fx = fx[0:fx.shape[0]-1, 0:fx.shape[1]-1]
    fy = fy[0:fy.shape[0]-1, 0:fy.shape[1]-1];
    ft = ft[0:ft.shape[0]-1, 0:ft.shape[1]-1];
    return fx, fy, ft

if __name__ == "__main__":
    im1 = np.asarray(Image.open('flow1-bw-0.png'))
    im2 = np.asarray(Image.open("flow2-bw-0.png"))
    fx, fy, ft = deriv(im1, im2)

import matplotlib.pyplot as plt
import numpy as np
import scipy.signal as si
from PIL import Image
import deriv
import numpy.linalg as lin

def lk(im1, im2, i, j, window_size) :
    fx, fy, ft = deriv.deriv(im1, im2)
    halfWindow = np.floor(window_size/2)
    curFx = fx[i-halfWindow-1:i+halfWindow,
               j-halfWindow-1:j+halfWindow]
    curFy = fy[i-halfWindow-1:i+halfWindow,
               j-halfWindow-1:j+halfWindow]
    curFt = ft[i-halfWindow-1:i+halfWindow,
               j-halfWindow-1:j+halfWindow]
    curFx = curFx.T
    curFy = curFy.T
    curFt = curFt.T

    curFx = curFx.flatten(order='F')
    curFy = curFy.flatten(order='F')
    curFt = -curFt.flatten(order='F')

    A = np.vstack((curFx, curFy)).T
    U = np.dot(np.dot(lin.pinv(np.dot(A.T,A)),A.T),curFt)
    return U[0], U[1]

```

```

if __name__ == "__main__":
    x=165
    y=95
    win=50
    im1 = np.asarray(Image.open('flow1-bw-0.png'))
    print im1.shape
    #im2 = np.asarray(Image.open('flow2-bw-0.png'))
    #im2 = np.asarray(Image.open('upright.png'))
    im2 = np.asarray(Image.open('dleft.png'))
    print im2.shape
    u, v = lk(im1, im2, x, y, win)
    print u, v
    plt.imshow(im1, cmap='gray')
    plt.hold(True)
    plt.plot(x,y, '+r');
    plt.plot(x+u*3,y+v*3, 'og')
    plt.show()

```

Kaynaklar

R. Collins Ders Notlari, www.cse.psu.edu/~rcollins/CSE486