KALMAN FİLTRESİ İLEÇERÇEVELERİN MERKEZ NOKTALARINI TAHMİN ETME

Kalman filtresi, fiziksel bir modele ve gürültülü ölçümlere dayalı tahminler kullanarak bir sistemin durumu hakkındaki bilgileri birleştiren bir algoritmadır. Ölçüm gürültüsünü filtrelediği için "filtre" olarak adlandırılır. GPS sensörlerinden ve radarlardan gelen bilgileri filtrelemek için yaygın olarak kullanılır.

Kalman algoritması yinelemelidir. Her adım tahmin ve ölçüm güncellemesi olmak üzere iki aşamadan oluşur. Tahmin aşaması, önceki durum tahmini ve işlem gürültüsüne dayalı olarak bir sonraki zaman adımında sistem durumu tahmininin bir hesaplamasıdır.

Daha sonra, sistemin durumuna bağlı olan bazı değişkenleri ölçüyoruz ve durum tahminini ölçümden elde edilen bilgilerle güncelliyoruz. Ölçümde de hata payı vardır, bu nedenle ölçüm gürültüsünü dikkate almamız gerekir. Bir durum tahminini güncellediğimizde, tahmin ve ölçümden gelen bilgileri birleştiririz.

sıralı_grafik_çizim dokümanına ek olarak;

1) Çerçeve konumlarını tahmin etmek (x ve P matrislerini güncellemek) için kalman fonksiyonunu oluşturuyoruz.

x-> durum matrisini temsil eder.

P -> belirsizlik kovaryans matrisidir.

```
def kalman(x, P, measurement, R, Q, F, H):

    """
    Parametreler:
     x: başlangıç durumu
     P: belirsizlik kovaryans matrisi
     measurement: gözlenen (ölçülen) konum
     R: ölçüm gürültüsü
     Q: hareket gürültüsü
     F: sonraki durum işlevi
     H:ölçüm fonksiyonu

     Return: (x, P) için güncellenmiş ve tahmin edilen yeni değerler
     """
    # x'i güncelleme
    y = np.array(measurement).T - np.dot(H,x) # ölçülen ve geçerli konum
arasındaki fark
    S = H.dot(P).dot(H.T) + R  # artık kovaryans
    K = np.dot((P.dot(H.T)), np.linalg.pinv(S))
    x = x + K.dot(y)
    I = np.array(np.eye(F.shape[0]))  # Kimlik matrisi
    P = np.dot((I - np.dot(K,H)),P)

    # Tahmin (x, P)
    x = np.dot(F,x)
    P = F.dot(P).dot(F.T) + Q
```

```
    return x, P
```

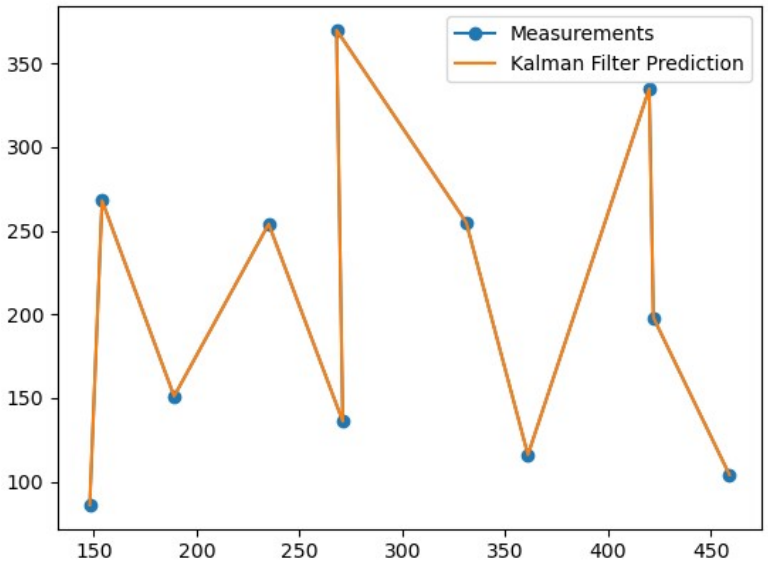
2) Tahmin edilen konumları elde etmek ve gerçek, tahmin edilen konumlara göre grafik çizmek için example fonksiyonu oluşturuyoruz.

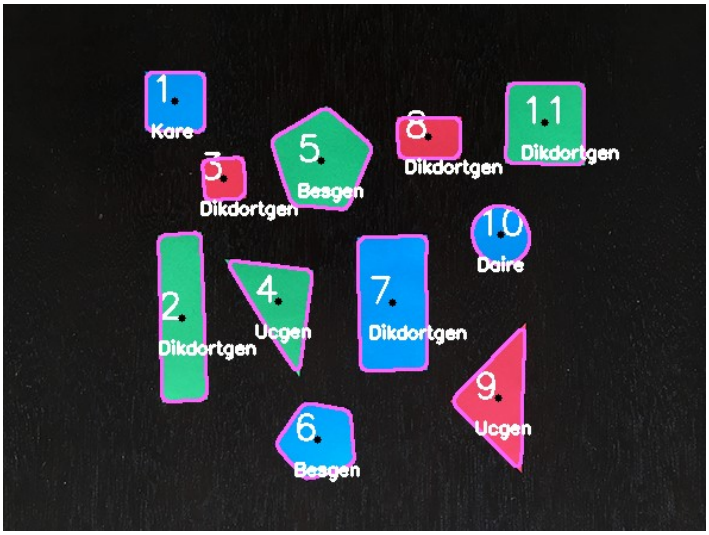
```
def example():
    F = np.array(np.eye(4)) # Sonraki durum işlevi
    Q = np.array(np.eye(4)) # hareket gürültüsü
    H = np.array([[1.0, 0.0, 0.0, 0.0],  # ölçüm fonksiyonu
                  [0.0, 1.0, 0.0, 0.0]])
    print("px",px)
    print("py",py)
    x = np.array([0.0, 0.0, 0.0, 0.0]).T # ilk durum
    P = np.array(np.eye(4))*1000 # ilk belirsizlik
    print("x",x)
    print("P",P)
    result = []
    R = 0.01**2
    # Gerçek kordinatların üzerinde döngü oluşturarak tahmin fonksiyonuna
gönderiyoruz.
    for meas in zip(px, py):
        x, P = kalman(x, P, meas, R, Q, F, H)
        result.append((x[:2]).tolist())
   # Tahmin edilen kordinatları x ve y eksenine ayrıştırıyoruz.
    kalman_x, kalman_y = zip(*result)
    # print("result",result)

    # Gerçek kordinatlara göre grafik çizdiriyoruz.
    plt.plot(px, py,marker = "o", label='Measurements')
    # Tahmin edilen kordinatlara göre grafik çizdiriyoruz.
    plt.plot(kalman_x, kalman_y, label='Kalman Filter Prediction')
    plt.legend()
    plt.show()
```

3) Kodu çalıştırmak için komut satırında aşağıdaki komutu çalıştırıyoruz.

{ python kalman.py --image im_shape.png --method "right-to-left" }

NOT: Kırmızı devamlı çizginin köşe noktaları gerçek kordinatları, mavi noktalar ise tahmin edilen kordinatları temsil etmektedir.





Sistem çıktısı aşağıdaki gibidir:

C:\Users\yasin\PycharmProjects\denme>python kalman.py --image im_shape.png --method "right-to-left"

px [148, 154, 189, 235, 271, 268, 331, 361, 420, 422, 459]

py [86, 268, 151, 254, 136, 370, 255, 116, 335, 198, 104]

kalmanx (147.99997660000474, 153.9812075123143, 189.0081929646442, 234.98510759495952, 271.00819374434883, 267.9769108734763, 331.0051933050603, 361.01089668042454, 419.9722132929079, 422.01348904757884, 459.0057004178703)

kalmany (85.99997660000469, 267.9812075123143, 151.00819296464414, 253.98510759495952, 136.0081937443489, 369.9769108734763, 255.00519330506032, 116.01089668042454, 334.9722132929079, 198.01348904757887, 104.00570041787029)