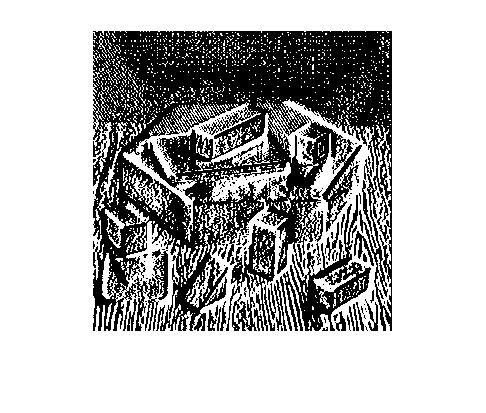
Soru1) Ödevin ilk kısmında Harris Corner Detection algoritmasını implement etmemiz gerekiyordu.

HarrisCornerDetection.m dosyasında şunları yaptık: Öncelikle kenarları tespit edeceğimiz resmi okuduk, sonrasında ise işlem kolaylığı sağlamak açısından 300x300’e scale ettik. Sonrasında ise, bu resmi harrisDetect() fonksiyonuna gönderdik. Bu fonksiyon bize iki adet resim dönüyor. Birinci resme sadece 1 filtre uygulanmış, ikinci resme ise 3 filtre birden uygulanmış. Sonrasında ise response dönen değerlere kırmızı bir kare çizdiriyoruz.

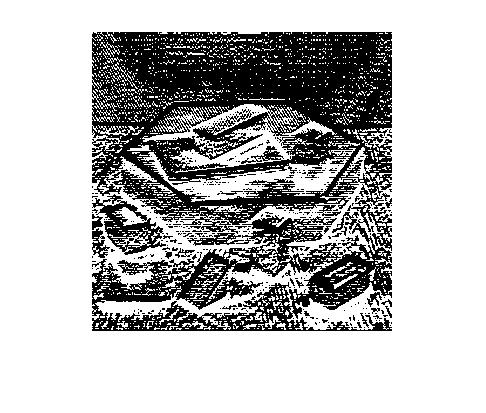
harrisDetect() fonksiyonu ise şunları yapıyor: Önce gelen resmi, eğer gray scale değilse, gray scale’a çeviriyor. Daha sonra bütün filtreleri sırayla oluşturuyor. Bunlardan birini seçip Single Filter Response olarak ilk out parametresi seçiyoruz. Daha sonra ise 3 filtre response matrisini alıyoruz, bunlardan herhangi birinde kenar olarak tespit edilmiş bir nokta varsa, bu noktayı Multiple Filter Response matrisinde de kenar olarak işaretliyoruz.

ApplyFilter1() fonksiyonu ise oluşturduğumuz x yönünde ve y yönünde uygulanacak filtreleri resme uyguluyor.

ApplyFilter2() fonksiyonu ise window function ile M matrisini çarparak Mc değerlerini oluşturuyor. Daha sonra bu Mc değerlerini bir tresholddan geçiriyoruz, son olarak da non-maximum suppression uygulayarak response matrisini oluşturuyoruz.

İlk aşama olarak resme 1x3lük filtreyi uyguladık ve aşağıdaki sonucu elde ettik.

Sonrasında ise 3x1’lik filtreyi uyguladık ve şu sonucu elde ettik.



Bu sonuçlar Single Response sonuçları.

Tresholddan sonra elimizde kalan sonuçlara bakalım:

Öncelikle 1x3-3x1’i tresholddan geçiriyoruz. Sonuç:



Sonrasında ise 1x5-5x1’i eşik değerinden geçiriyoruz.



Son olarak da 1x7-7x1’e bakalım.

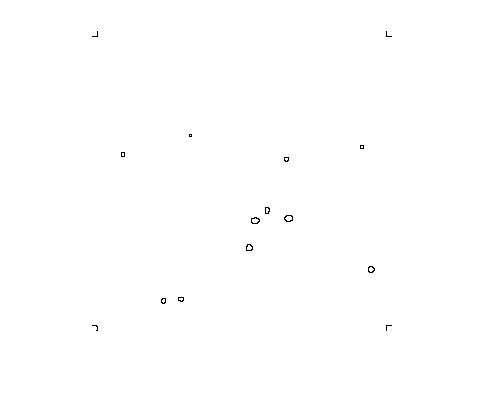


Sonrasında ise 1x5-5x1 ve 1x7-7x1 filtreleri de uyguluyoruz.

Üçünün toplam response değeri ise:

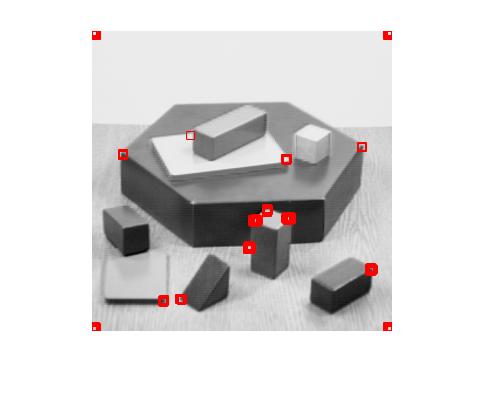


Bu değerlere non-maximum suppression uyguladıktan sonra response matrisi şu hale geldi:

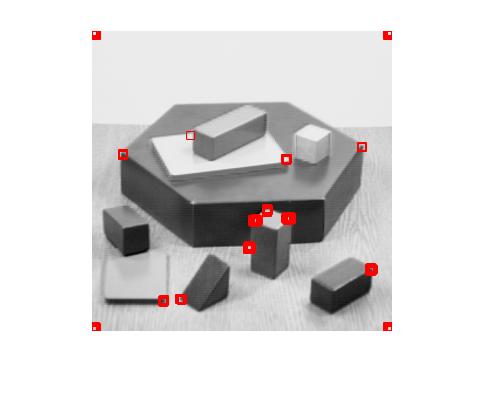


En son adım olarak kırmızı karelerimizi bastıralım.

Single Filter Response:

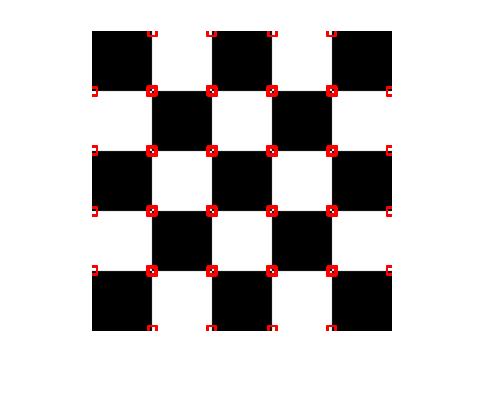


Multiple Result ise:



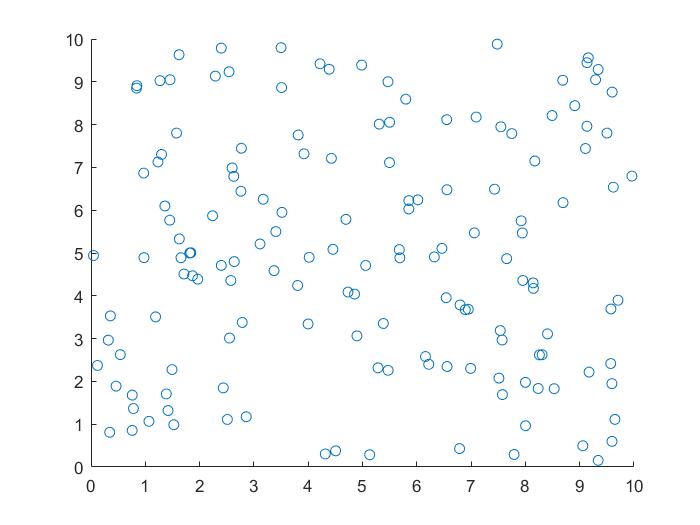
Başka bir resimdeki sonuca bakalım.

Satranç tahtası için sonuç:



Soru2) Ödevin ikinci kısmında ise Ransac algoritmasını uygulayacağız. Bu kısmın kodları sadece RansacMain.m adlı dosyada.

Öncelikle, ekrana üzerinde işlem yapacağımız noktaları oluşturuyoruz.

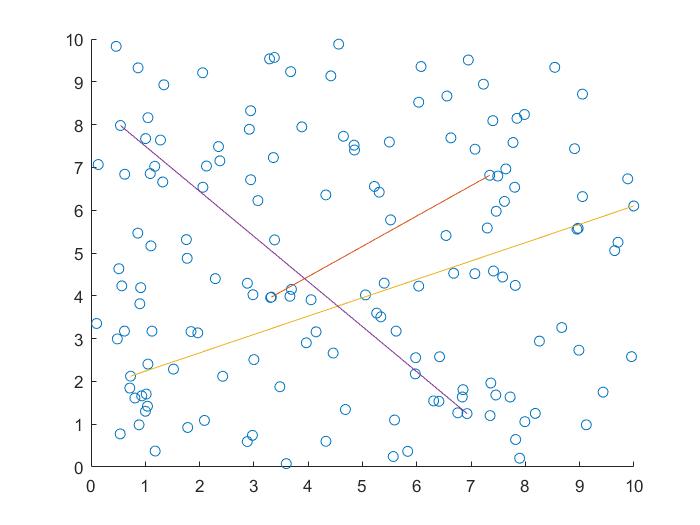


Sonraki adım olarak, rasgele ikili seçip, bu iki noktadan bir doğru oluşturuyoruz. Daha sonra diğer noktaların bu doğruya uzaklıklarını hesaplıyoruz.

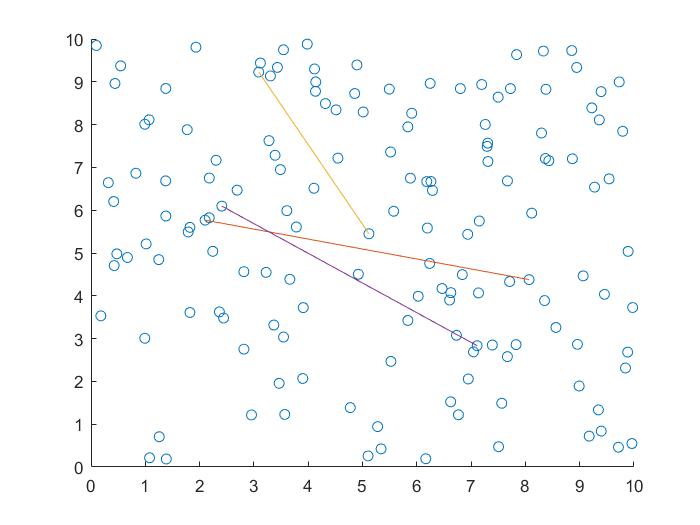
Bir matriste en küçük üç değeri saklıyoruz. Eğer bu matrise girecek bir değer varsa bu değer, bu matristeki elemanların en az birinden küçük olmalı. O nedenle her seferinde matrisin en büyük elemanı ile yeni elemanları karşılaştırıyoruz. Matristen atacağımız eleman matrisin en büyük değerine sahip olan eleman olmalı. Bu şekilde, bütün noktaları gezince işlemimiz bitmiş oluyor. Bütün noktaları hesapladıktan sonra tekrar başa dönüp en küçük 3’ünü seçmemiz gerekmiyor. Zaten bu işlem için bir de sıralama maliyeti vardı. O maliyetten de kurtulmuş oluyoruz.

Sonuçta elde ettiğimiz doğruları ekrana basıyoruz.

Birinci çıktı:



İkinci çıktı:



Üçüncü değer grubu için çıktı ise:

