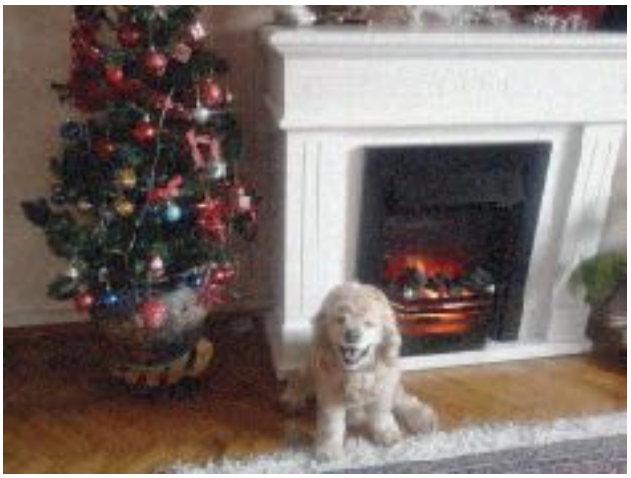
1. Aşağıdaki işlemleri yapan MatLab /C++/OpenCV/Python kodlarını yazınız. Yazdığınız ortanca süzgeçleme rutini hazır süzgeçleme komutu kullanmamalıdır. Görüntüden okuma, maske içerisindeki piksel gri düzeylerini sıralayarak ortanca değeri maskenin merkezindeki piksele atayacak şekilde maskeleme yapıyor olmalıdır.

a-) noisy_foto.jpeg renkli görüntü dosyasını okuyunuz ve 8 bit gri düzey görüntüye dönüştürerek noisy_gray.jpeg dosyasına yazdırınız.

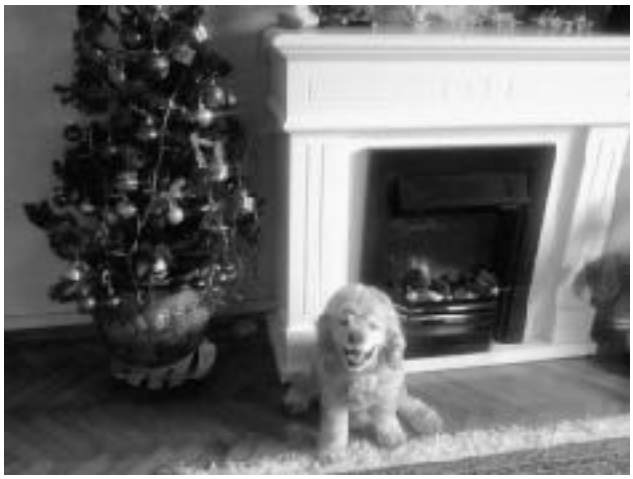


noisy_foto.jpg

noisy_gray.jpg

b-) noisy_gray.jpeg dosyasını okuyunuz ve 7x7 maske boyutlu ortanca süzgeçleme (median filtering) uygulayarak çıkışı median1_gray.jpeg dosyasına yazdırınız.

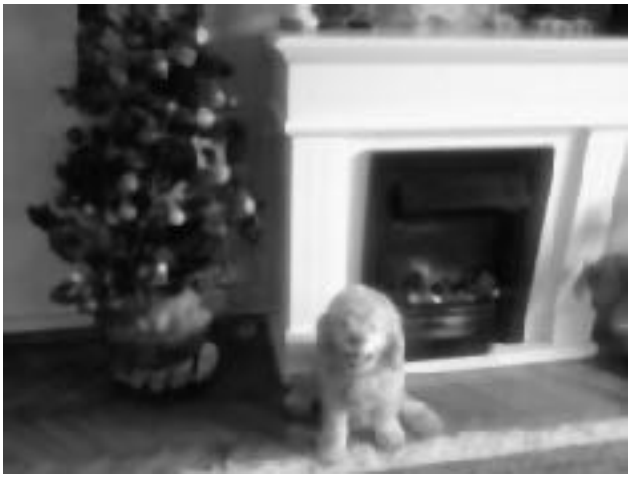
noisy_gray.jpg



median1_gray.jpg

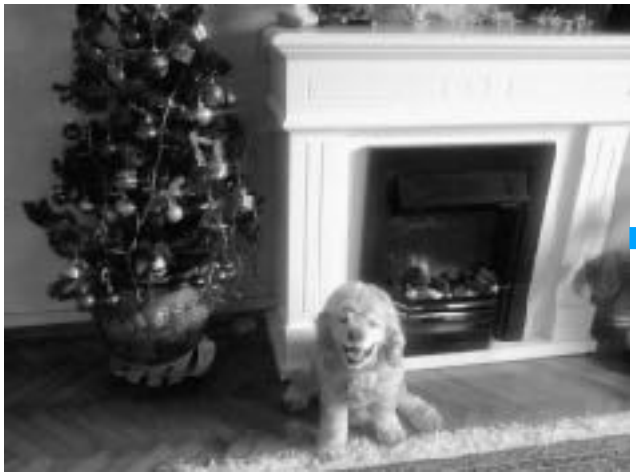
c-) (b)'deki işlemi 27x27 lik maske için tekrarlayınız ve sonucu median2_gray.jpeg dosyasına yazdırınız.

noisy_gray.jpg

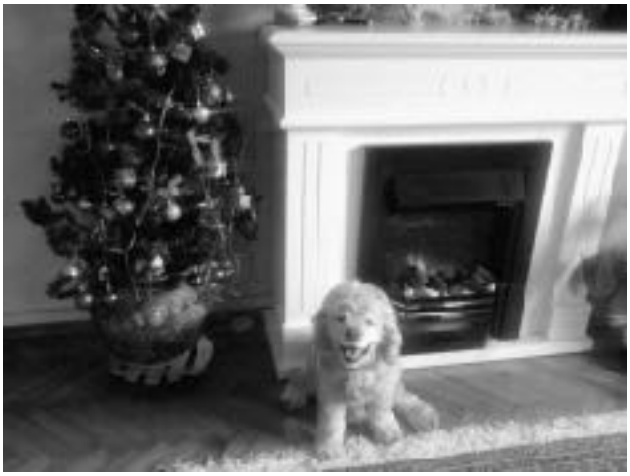


median2_gray.jpg

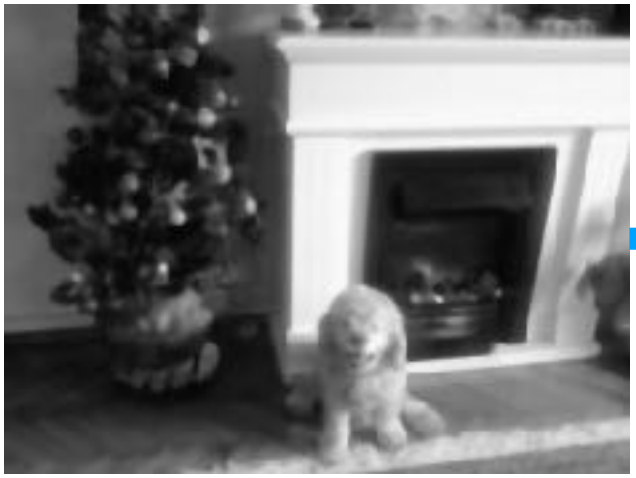
To make test the written algorithm for median filter, the opencv medianfilter function was ran. I compare the results of "my algorithm" and "OpenCV Algorithm"



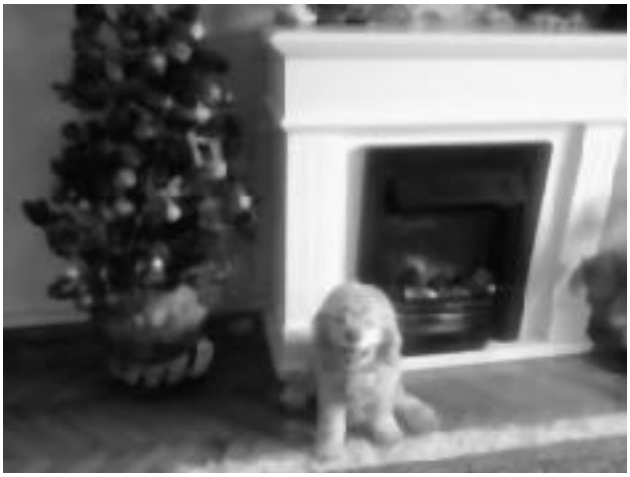
median1_gray.jpg



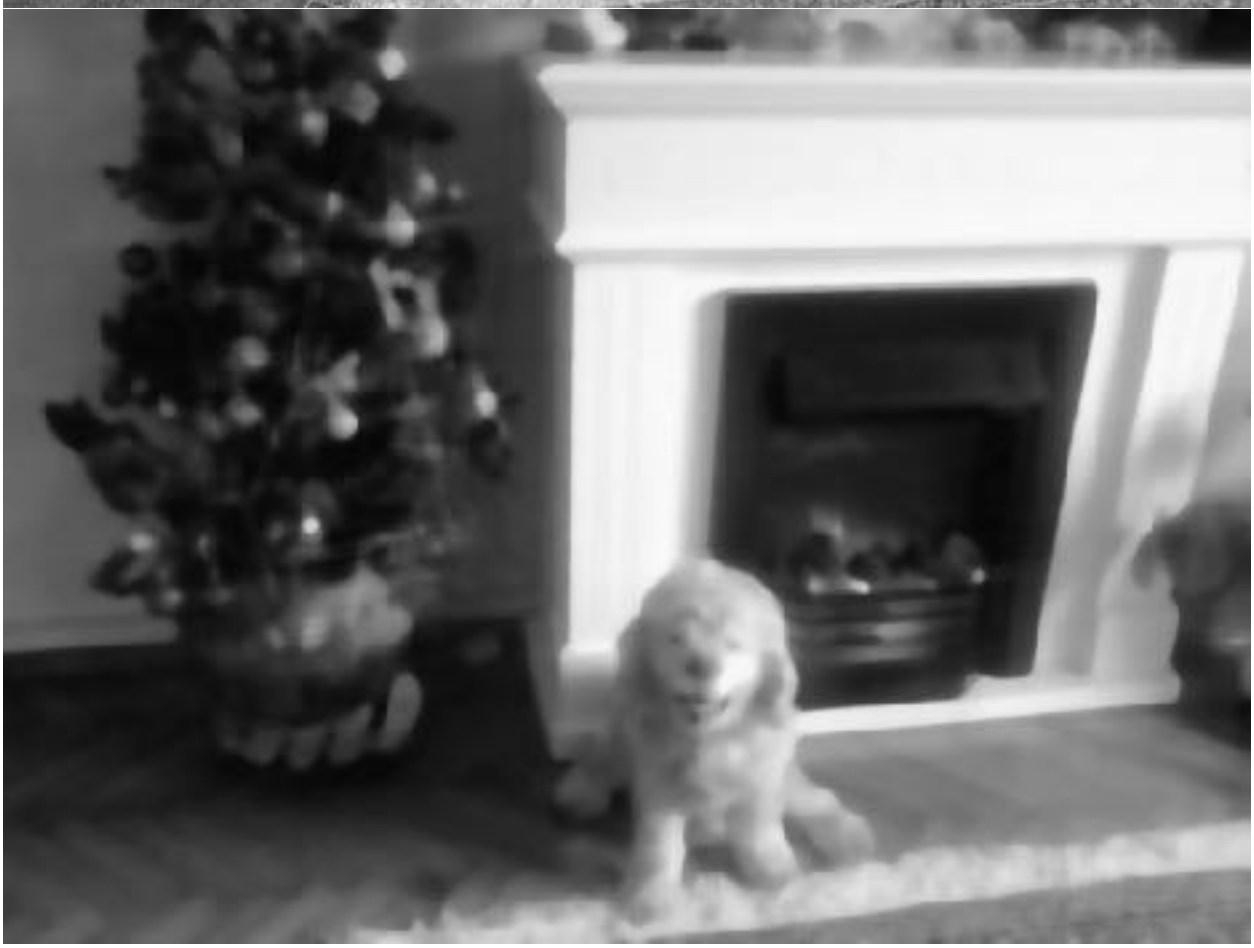
opencv_median1_gray.jpg



median2_gray.jpg



opencv_median2_gray.jpg



d-)(b) ve (c)'de bulduğunuz sonuçları nedenlerini de yazarak yorumlayınız.

B şıkkında 7*7 kernel boyutu kullanılarak elde edilen sonuç'ta "Salt-and-pepper" gürültülü resimdeki gürültülerin başarılı bir biçimde elimine edildiği görülmektedir.

C şıkkında ise 27*27 kernel boyutlu median filter kullanılmıştır. Bu filtre sonucu "Salt-and-pepper" gürültüsü giderilmiştir fakat resim bulanıklaştığı görülmektedir.

Median filtre kullanılarak piksek değeri yüksek yada piksel değeri düşük gürültüler başarılı bir şekilde elimine edilebilir. Convolusyon sırasında kernel boyutunca elde edilen piksel değerleri sıralandığı takdirde orta sıradaki (median) değerin yeni değer olarak belirlendiği Median Filter'da, yüksek değerli bir gürültü sıralamanın sonunda; düşük değerli bir gürültü sıralamanın başında yer alacaktır. Bu yüzden gürültü elimine edilecektir.

Zira, 7*7 ve 27*27 kernel boyutlu filtrelerde "Salt-and-pepper" gürültüsünün giderildiği görülmektedir.

Fakat 27*27 kernel boyutlu filtre kullanıldığında detaylar da elimine olduğu görülür. (Bakınız şömine üstündeki çember, şömine kenarları). Bu kernelde bir pikselin yeni değerinin bulunması için 27*27 =729 piksel sıralanır ve ortanca değeri bulunur. Yine "Salt-and-pepper" gürültüsü giderilir fakat yeni piksel değeri uzak çevresindeki piksellerden rastgele birisini alır. Bu durumda resim genel çerçevesini korur fakat artık daha bulanık bir çerçeve elde edilmiş olur. Kenar gibi geçiş noktalarındaki piksellerin sayısı az olduğundan dolayı ortanca değer olma olasığının az olur bu yüzden bu bilgiler yüksek ihtimalle kaybedilir. Gölge, parlaklığın az/çok olduğu yerlerde ise bunların piksel değerleri yüksek/yada düşük olacağı için ve 27*27=729 piksel arasında sıralandığı için bu bilgilerin de çoğu kaybolacaktır. (Bakınız çam ağacı gölgesi).

Sonuç olarak median filtresi ortanca değeri bularak atama yaptığı için uygun olan minumum kernel boyutu kullanılmalıdır.

d-)Sorting Algorithm

Sıralama algoritması ortanca değeri bulmak için kullanıldığı için ortanca değerden sonraki değerlerin sıralanması gerek yoktur.

Küçükten büyüğe sıralayan Algoritma (Selection Sort) şu şekilde çalışmaktadır:
N (tek sayı) adet sayı olduğu düşünülürse
i: adım sayısı, ilk değer = 1;
Adım 1: Sıralı dizinin i'ninci elemanını bulmak için original dizinin i'ninci elemanı ve sonrası elemanlarına bak en küçüğü bul.

Adım 2: bulunan en küçük elemanı sıralı dizinin i.elemanına ata.

Adım 3: i'yi bir artır.

Adım 4: i Ortanca değer'e (N-1/2) küçük yada eşitse Adım 1'e git.

Böylelikle original dizi küçükten büyüğe ortanca değere kadar sıralanmış olur.

Yapılan karşılaştırma sayısı: $\frac{(N-1)!}{((N-1)/2-1)!}$