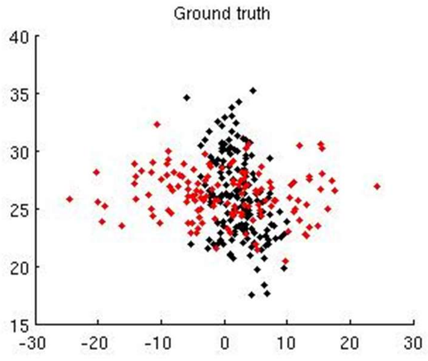
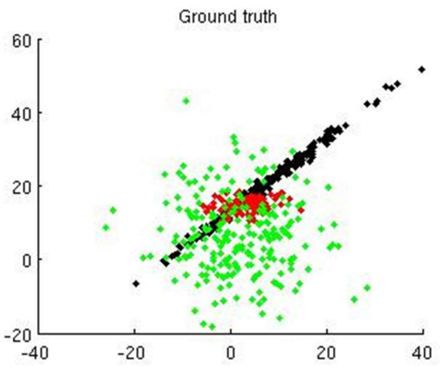
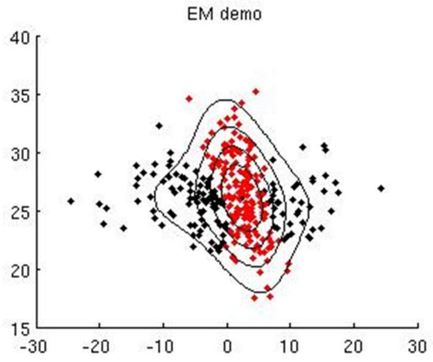
2.2



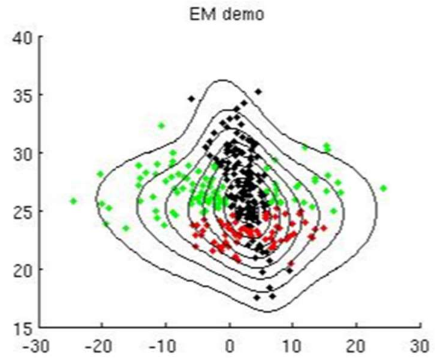
3-1 ground truth of 2 clusters



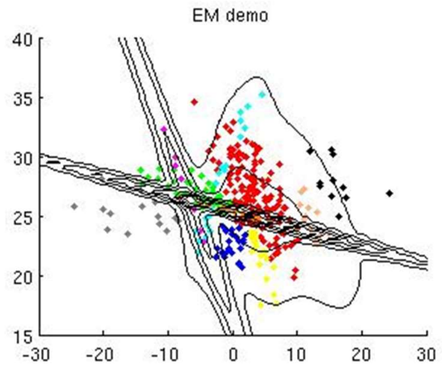
3-2 ground truth of 3 clusters



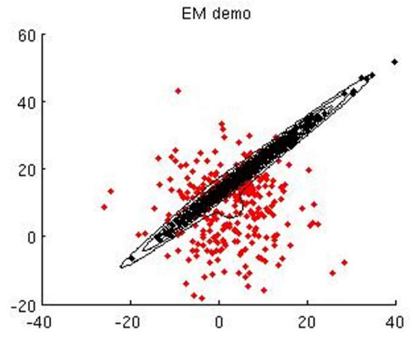
3-3 EM result with cluster num=2, guess num=2



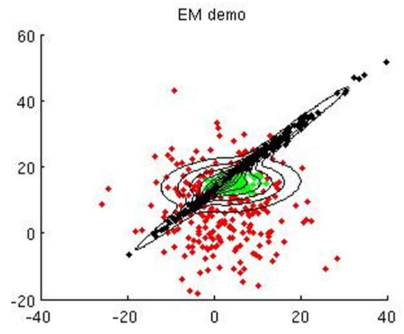
3-4 EM result with cluster num=2, guess num=3

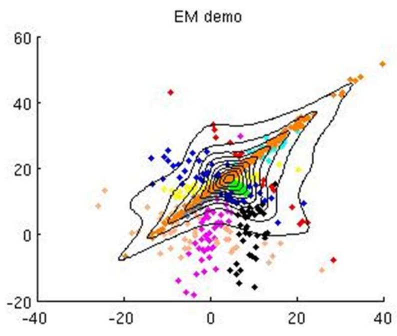


3-5 EM result with cluster num=2, guess num=10



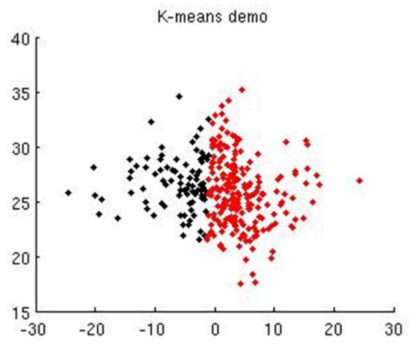
3-6 EM result with cluster num=3, guess num=2



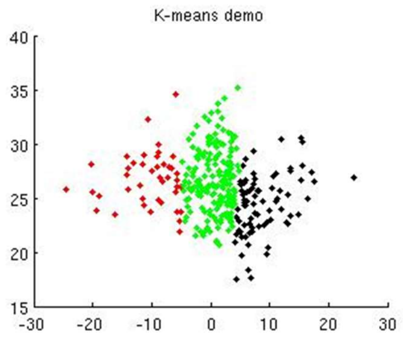


3-7 EM result with cluster num=3, guess num=3 3-8 EM result with cluster num=3, guess num=10

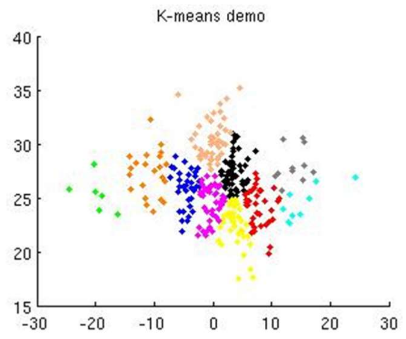
GMM gives satisfactory fit on data that also follows gaussian distribution, especially when the number of clusters we guess is close to actual one. In this case, we can easily merge 2 or 3 clusters into 1 actual cluster. As the number we guess goes far away from actual value, data that are less likely to be generated by a gaussian model will first to be split.



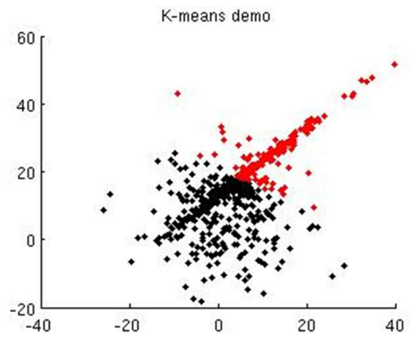
3-9 KMeans result with cluster num=2, guess num=2



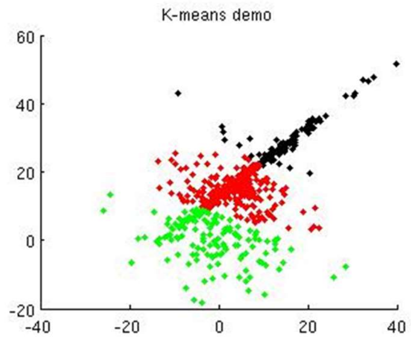
3-10 KMeans result with cluster num=2, guess num=3



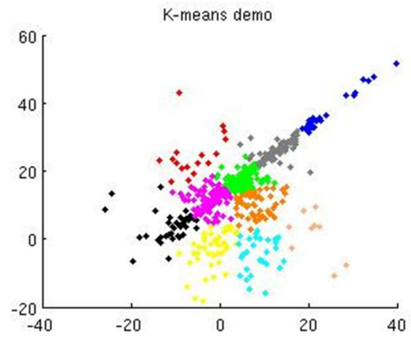
3-11 KMeans result with cluster num=2, guess num=10



3-12 KMeans result with cluster num=3, guess num=2

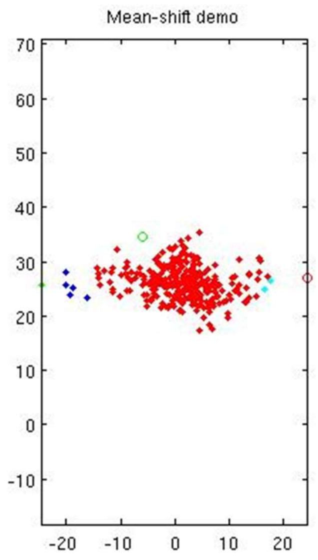


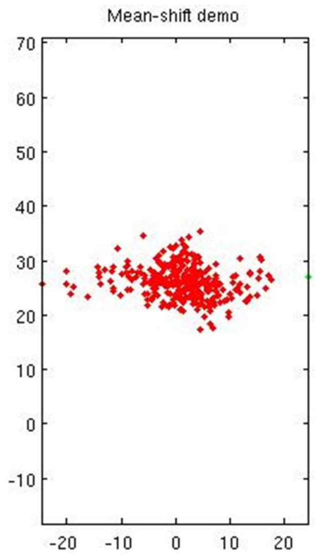
3-13 KMeans result with cluster num=3, guess num=3



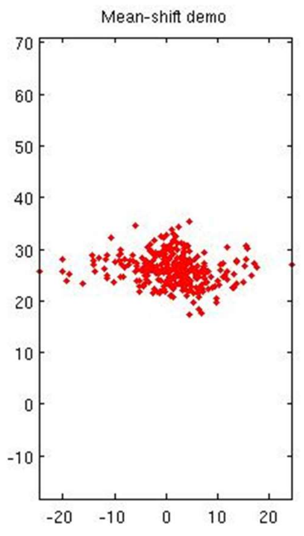
3-14 KMeans result with cluster num=3, guess num=10

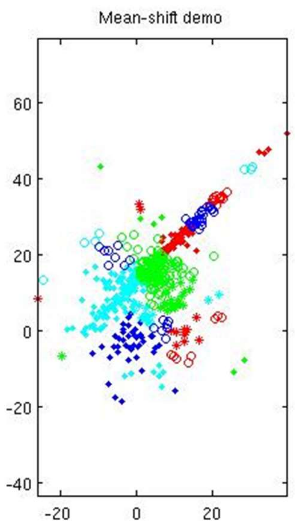
For KMeans clustering, since cluster assignment only depends on which center is nearer rather than which distribution it is more likely to be in, no overlapping will appear among clusters and thus it works poorly on clusters that have overlapping areas.



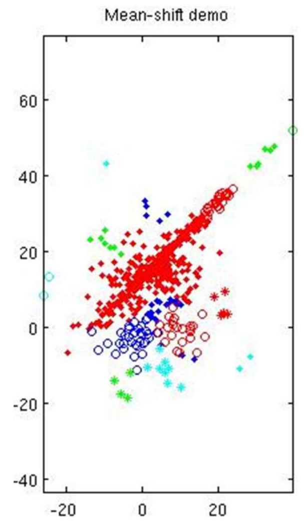


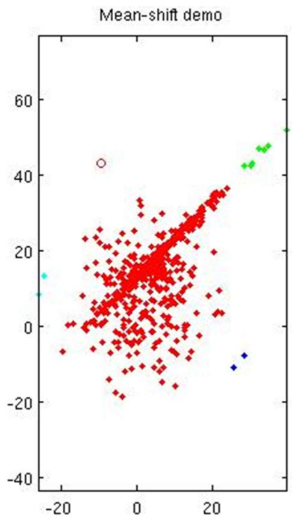
3-15 meanshift result with cluster num=2, win size=4 3-16 meanshift result with cluster num=2, win size=6





3-17 meanshift result with cluster num=2, win size=10 3-18 meanshift result with cluster num=3, win size=4

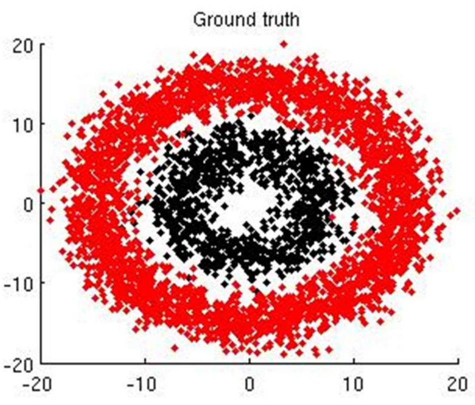


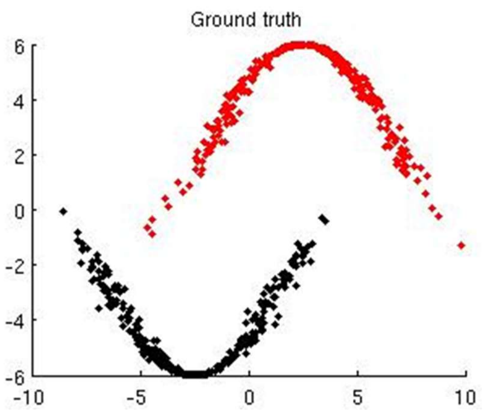


3-19 meanshift result with cluster num=2, win size=6 3-20 meanshift result with cluster num=3, win size=10

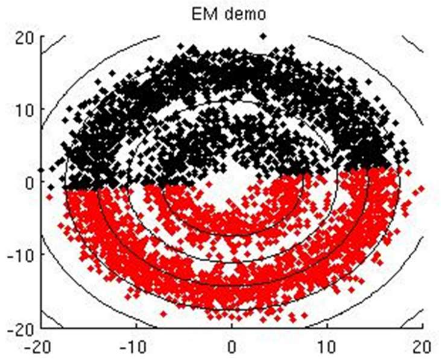
Mean shift clustering also don't cope well with overlapping clusters. Also, window size should be carefully chosen according to range of distribution of data. A window size too large will not only produce wrong results but also put virtually everything into 1 cluster.

2.3

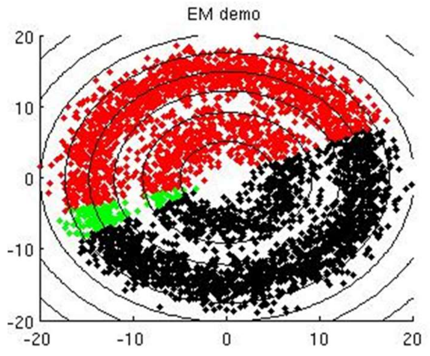




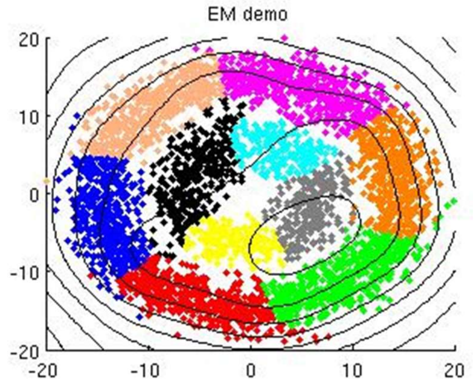
3-21 circular cluster ground truth 3-22 shaped cluster ground truth



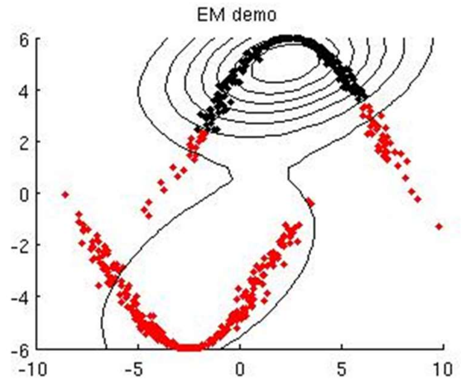
3-23 circular cluster EM cluster num=2



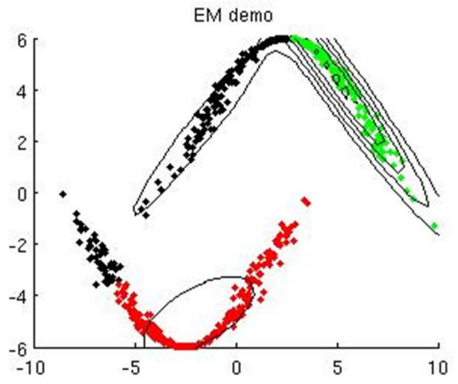
3-24 circular cluster EM cluster num=3



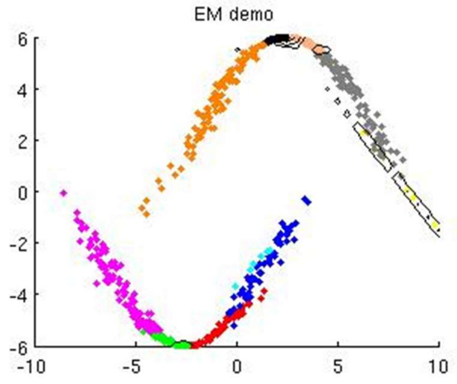
3-25 circular cluster EM cluster num=10



3-26 shaped cluster EM cluster num=2

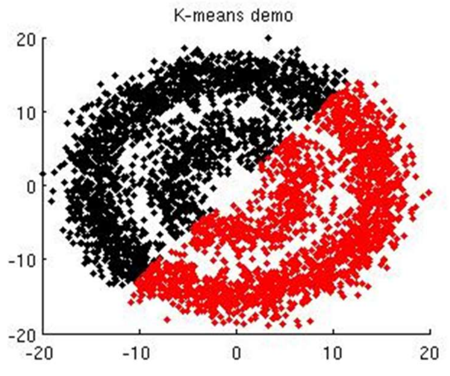


3-27 shaped cluster EM cluster num=3

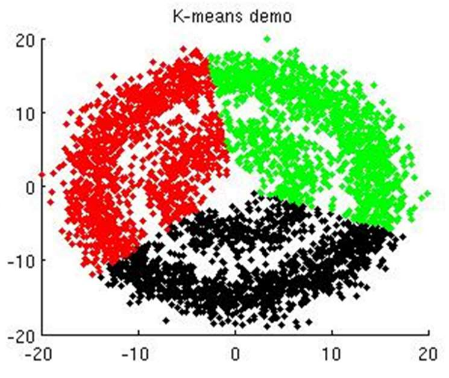


3-28 shaped cluster EM cluster num=10

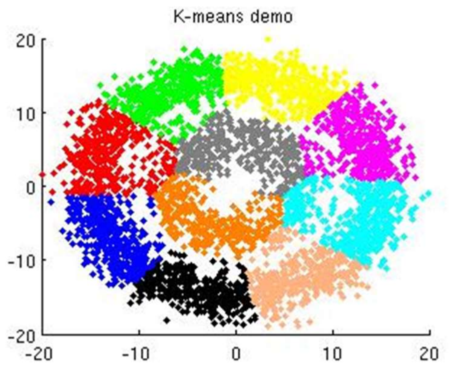
When number of predicted clusters is small, EM do not perform well on irregular clusters. Modeling ability of GMM heavily depends on number of gaussian distributions as it is highly likely that 1 gaussian distribution can model 1 part of the model, and the more the number of gaussians the better the model can be represented. As the number of gaussians increase, modeling ability of GMM indeed gets better, as we can see when there are 10 gaussians inner 4 forms one previous cluster and outer 6 forms the other.



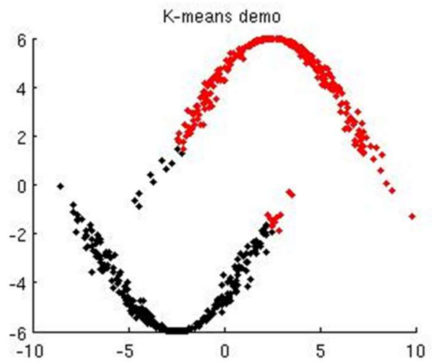
3-29 circular cluster KMeans cluster num=2



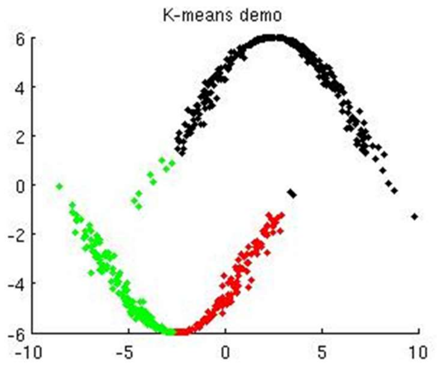
3-30 circular cluster KMeans cluster num=3



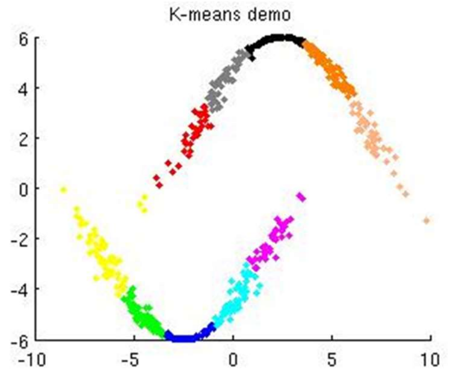
3-31 circular cluster KMeans cluster num=10



3-32 shaped cluster KMeans cluster num=2

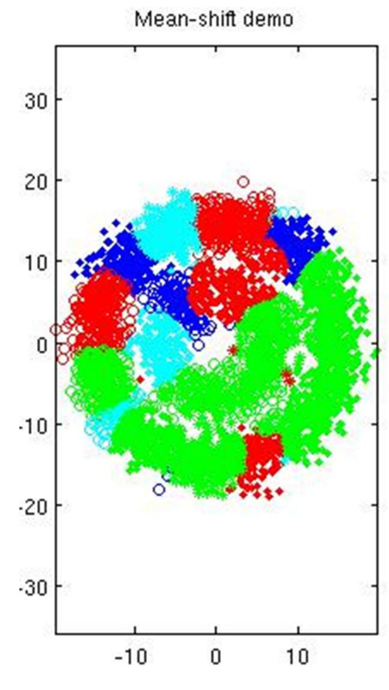


3-33 circular cluster KMeans cluster num=3

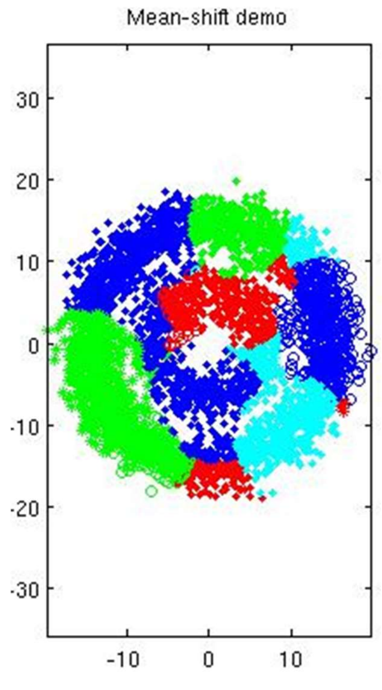


3-34 shaped cluster KMeans cluster num=10

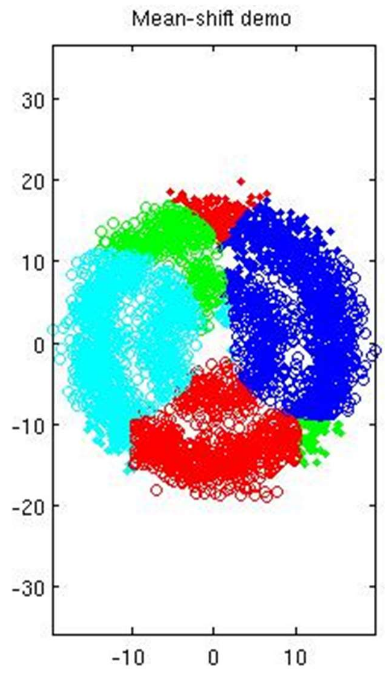
Similar case also goes for KMeans clustering, with bad performance when guessed number of clusters are low and performance increase with it. However, it still works a bit worse when number of clusters is the same as that of GMM.



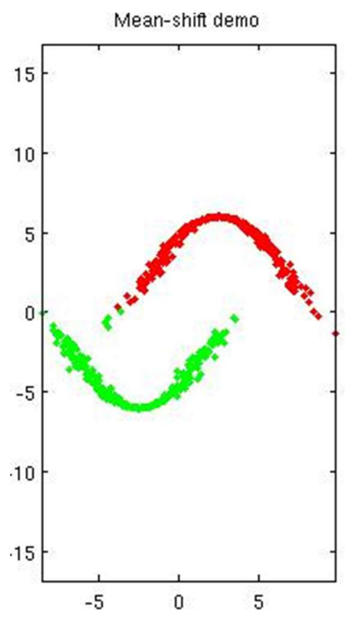
3-29 circular meanshift win size=4

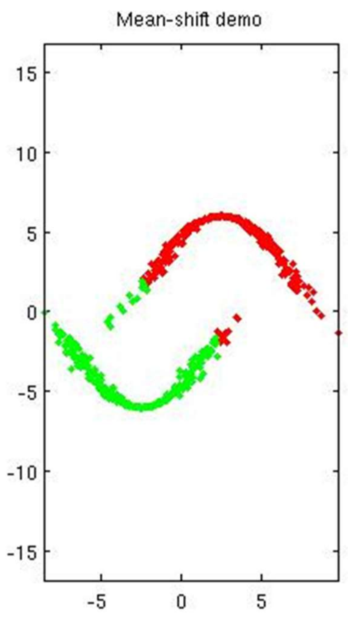


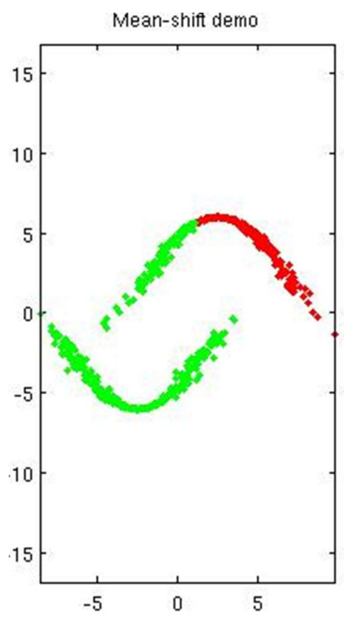
3-30 circular meanshift win size=6



3-31 circular meanshift win size=10





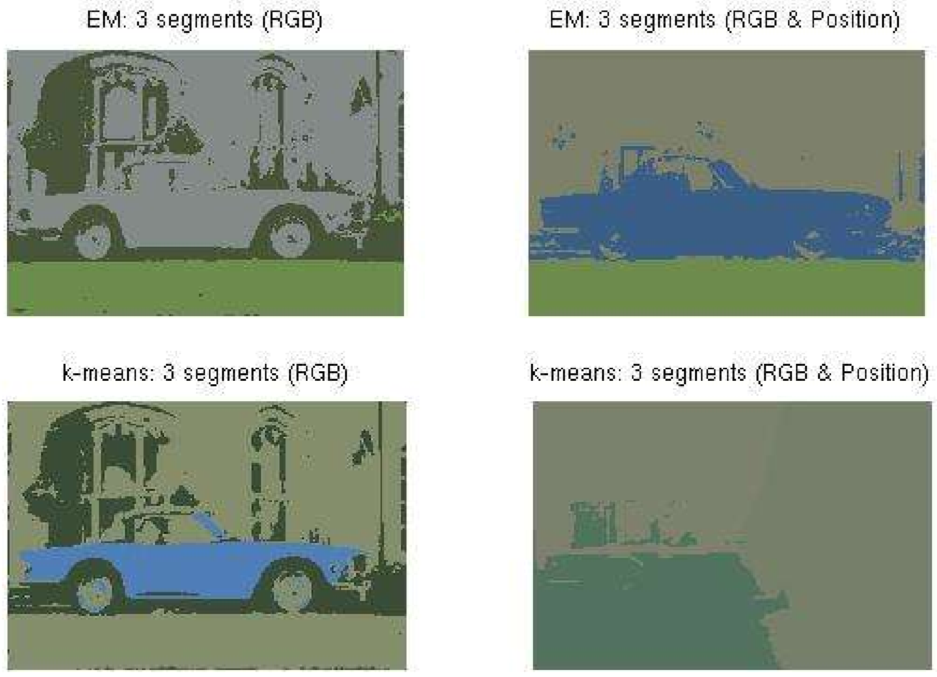


3-32 shaped meanshift win size=4 3-33 shaped meanshift win size=6 3-34 shaped meanshift win size=10

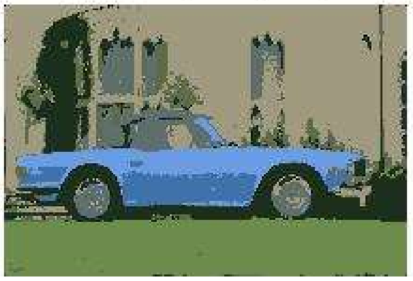
Performance of meanshift is unacceptable with circular clusters whatever window size is, but for shaped clusters results are pretty well if we control carefully the window size.

So when number of clusters is large, GMM and KMeans gain acceptable ability of modeling irregular data distribution, while meanshift will work well if distances between clusters are large so that windows will not be misguided.

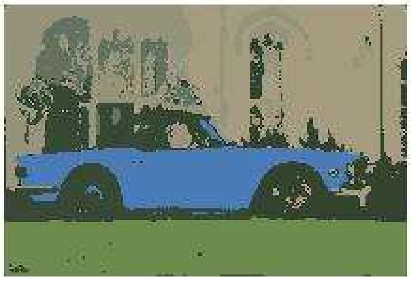
3.1


EM: 3 segments (RGB)
EM: 3 segments (RGB & Position)
k-means: 3 segments (RGB)
k-means: 3 segments (RGB & Position)

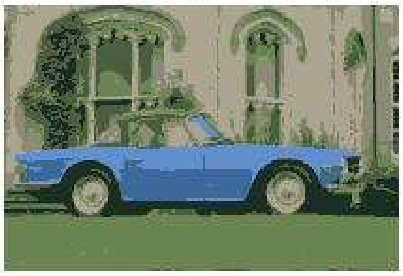
EM: 7 segments (RGB)



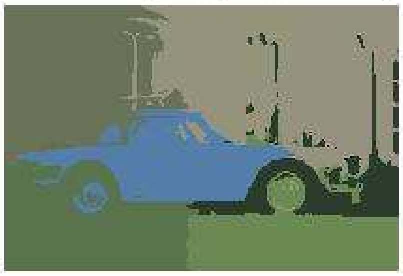
EM: 7 segments (RGB & Position)



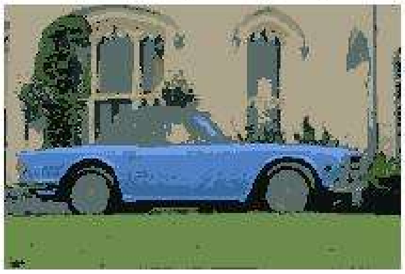
k-means: 7 segments (RGB)



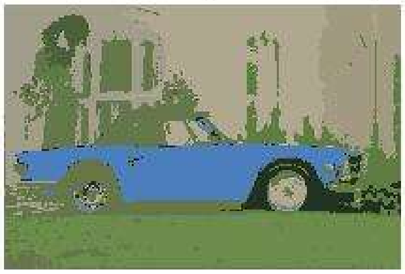
k-means: 7 segments (RGB & Position)



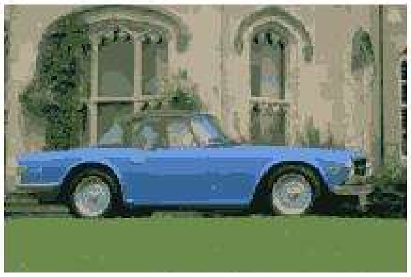
EM: 13 segments (RGB)



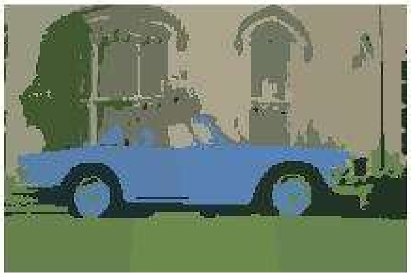
EM: 13 segments (RGB & Position)



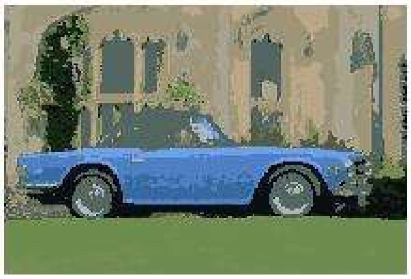
k-means: 13 segments (RGB)



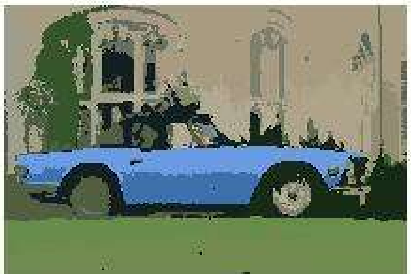
k-means: 13 segments (RGB & Position)



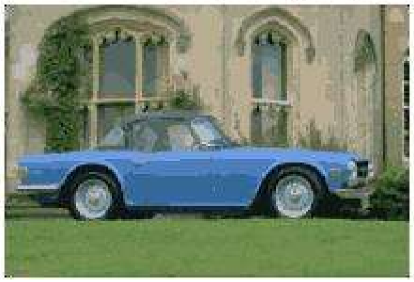
EM: 25 segments (RGB)



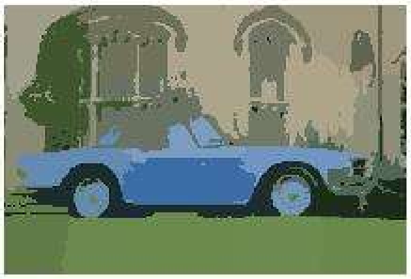
EM: 25 segments (RGB & Position)



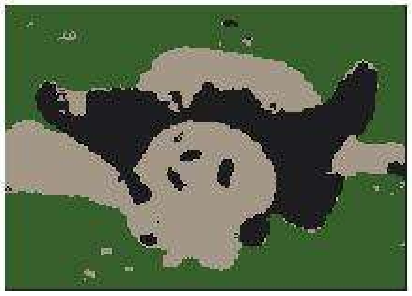
k-means: 25 segments (RGB)



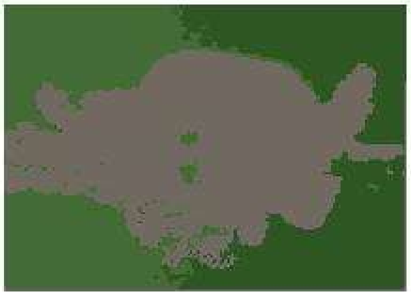
k-means: 25 segments (RGB & Position)



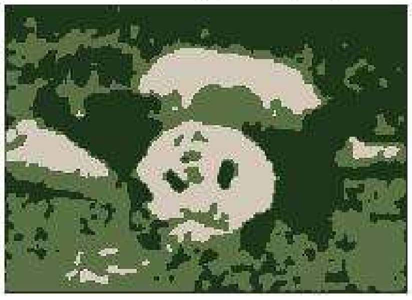
EM: 3 segments (RGB)



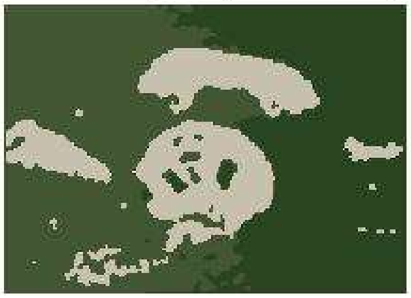
EM: 3 segments (RGB & Position)



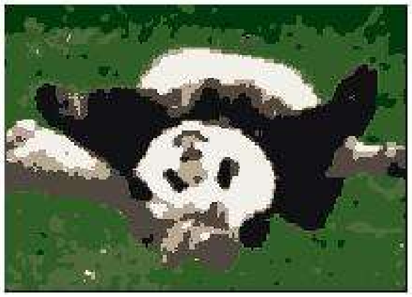
k-means: 3 segments (RGB)



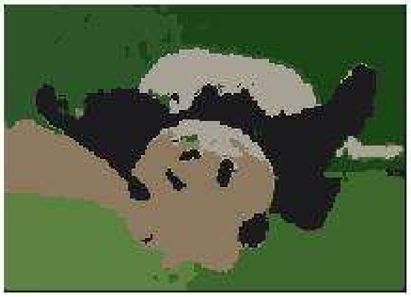
k-means: 3 segments (RGB & Position)



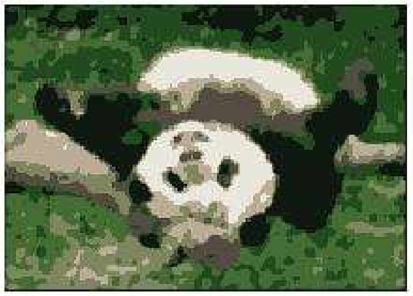
EM: 7 segments (RGB)



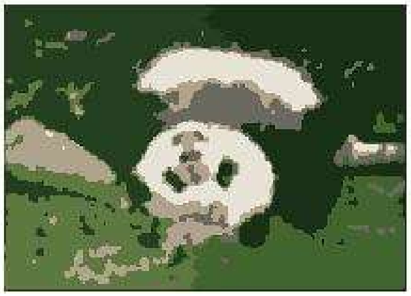
EM: 7 segments (RGB & Position)



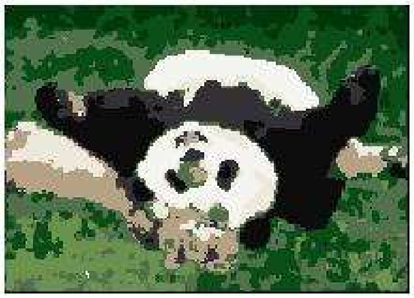
k-means: 7 segments (RGB)



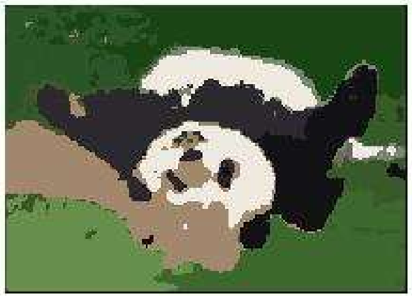
k-means: 7 segments (RGB & Position)



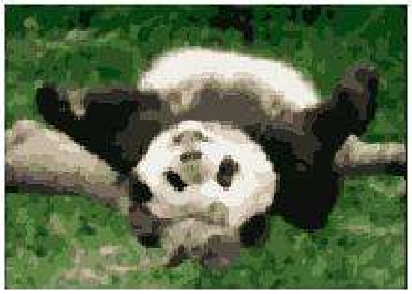
EM: 13 segments (RGB)



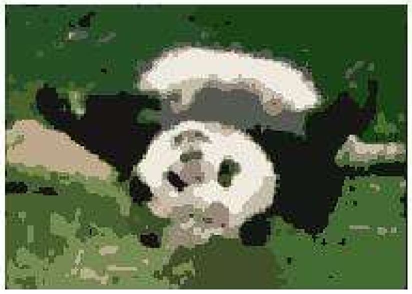
EM: 13 segments (RGB & Position)



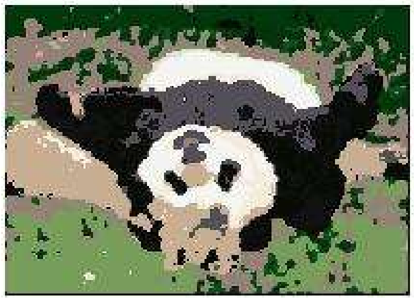
k-means: 13 segments (RGB)



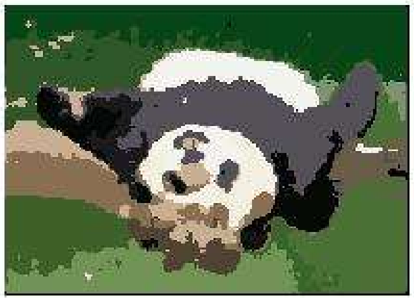
k-means: 13 segments (RGB & Position)



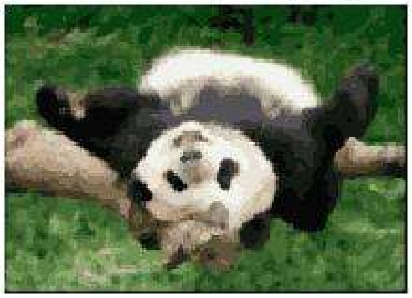
EM: 25 segments (RGB)



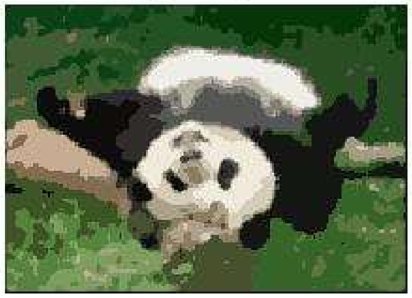
EM: 25 segments (RGB & Position)



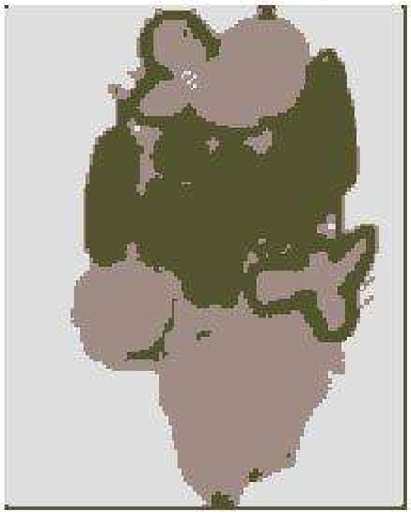
k-means: 25 segments (RGB)



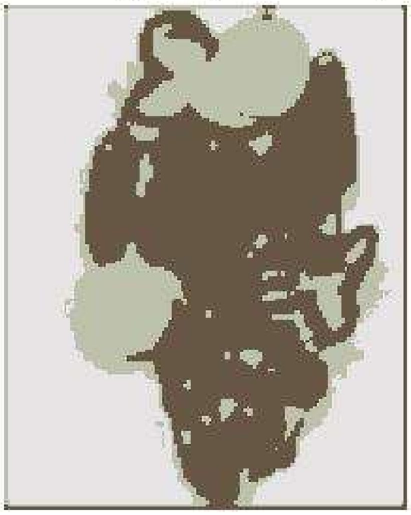
k-means: 25 segments (RGB & Position)



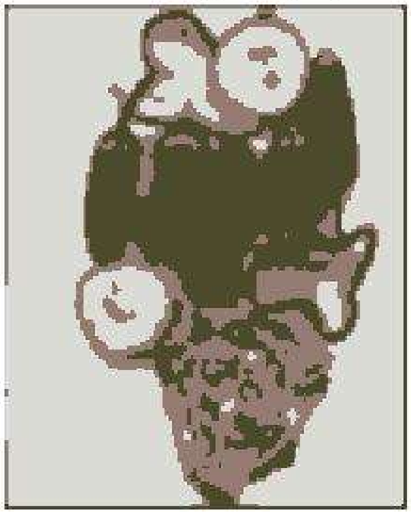
EM: 3 segments (RGB)



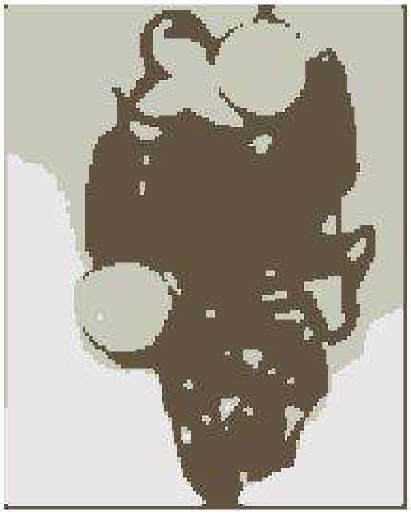
EM: 3 segments (RGB & Position)



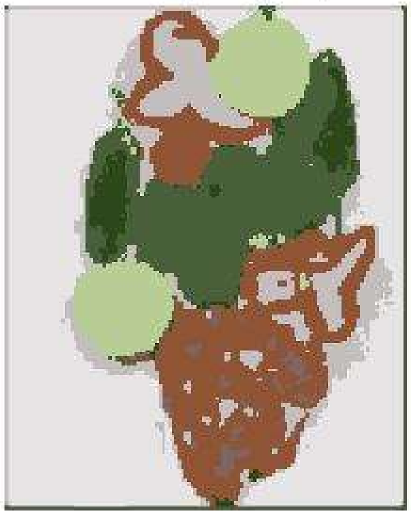
k-means: 3 segments (RGB)



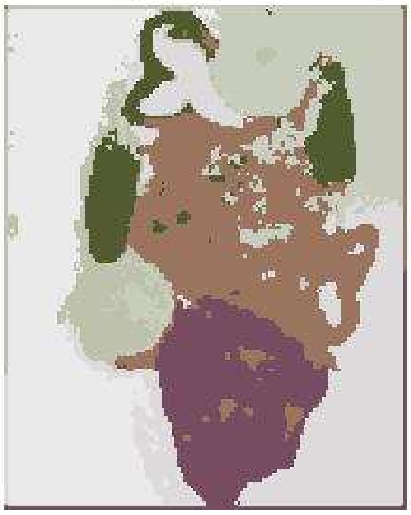
k-means: 3 segments (RGB & Position)



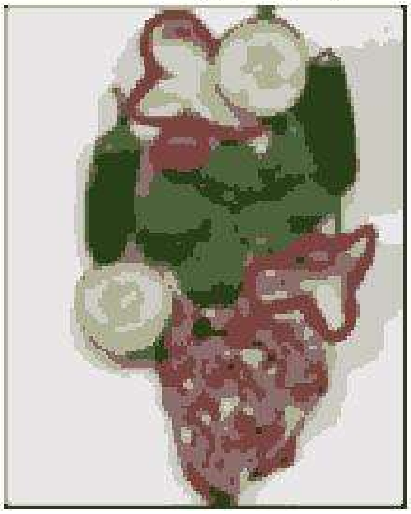
EM: 7 segments (RGB)



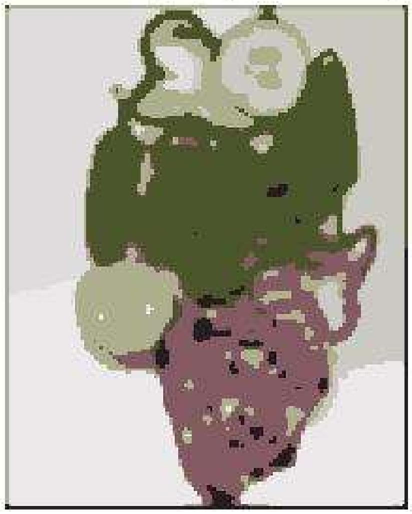
EM: 7 segments (RGB & Position)



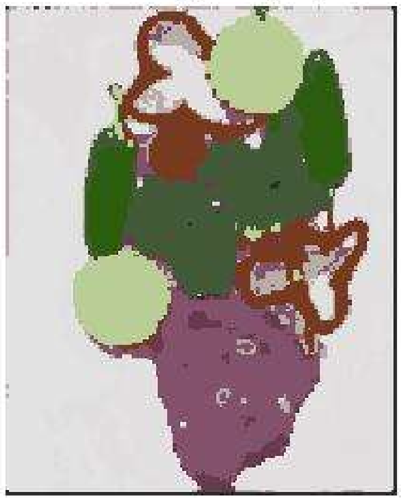
k-means: 7 segments (RGB)



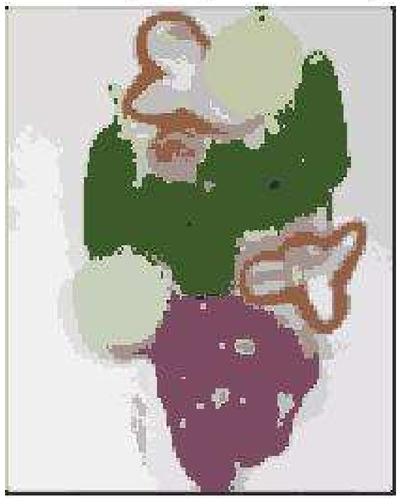
k-means: 7 segments (RGB & Position)



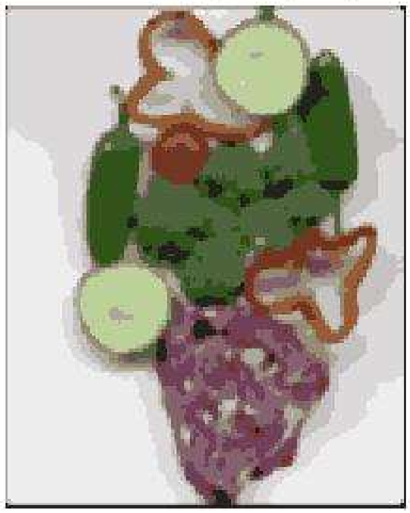
EM: 13 segments (RGB)



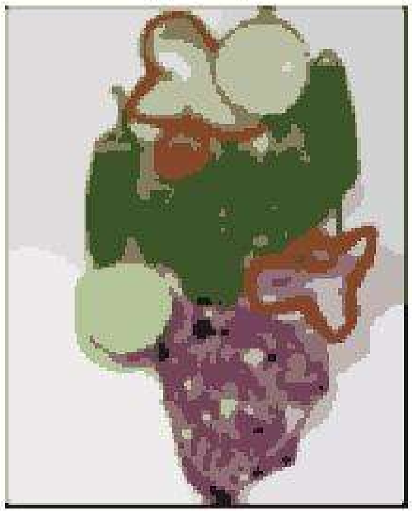
EM: 13 segments (RGB & Position)



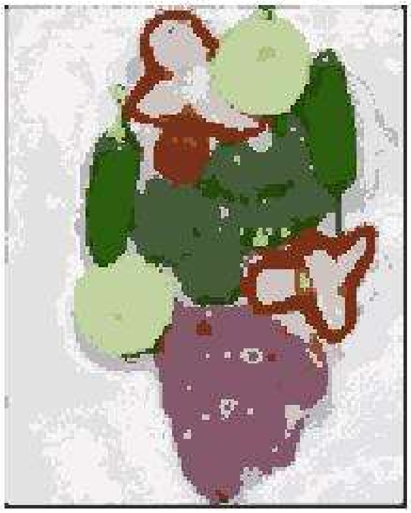
k-means: 13 segments (RGB)



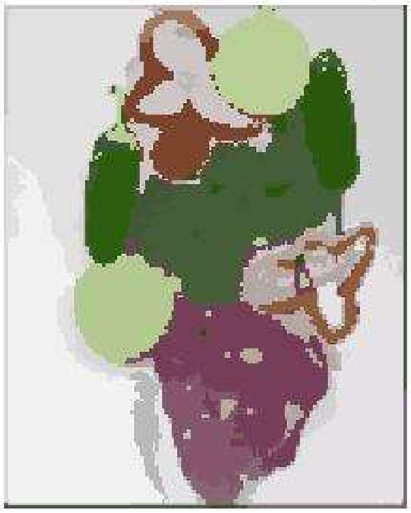
k-means: 13 segments (RGB & Position)



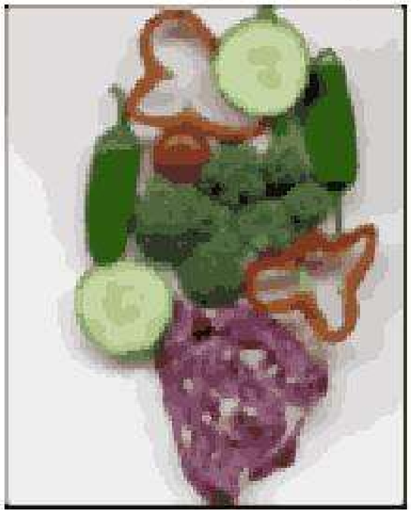
EM: 25 segments (RGB)



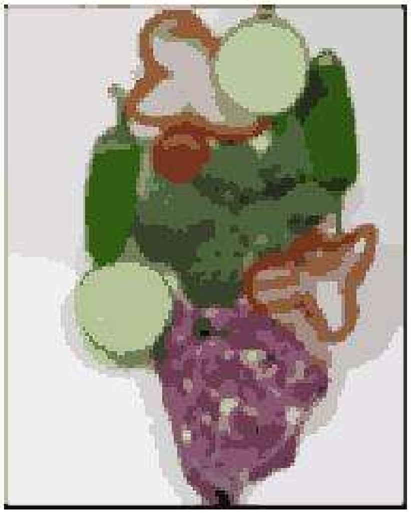
EM: 25 segments (RGB & Position)



k-means: 25 segments (RGB)

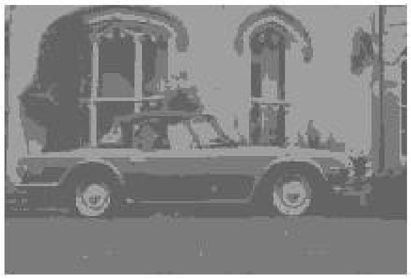


k-means: 25 segments (RGB & Position)

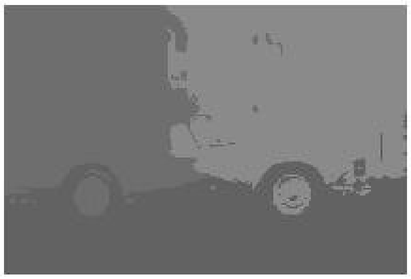


For colored images, increasing number of clusters always bring better performance when using KMeans and EM.
For KMeans using RGB feature, the result looks more and more alike the original picture.
EM with RGB is also acceptable, yet its performance is better on images with less details (veggis.png). For images that have color changes that not so obvious (yellow grass in panda.png) or many details (car.png), there are some minor false segmentations.
When using position as extra info, segmentation performance get worse compared to that using only RGB with same number of clusters: they still does well on segmentation on a large scale, but fails to detect detailed features. KMeans still performs better than EM regrading accuracy (rare wheel of car in car.png) and detail detection.

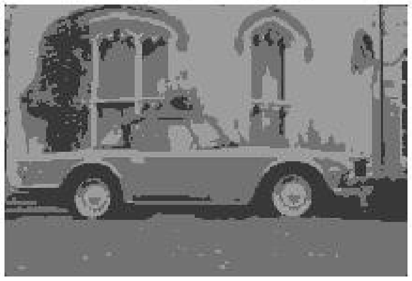
EM: 3 segments (Grayscale)



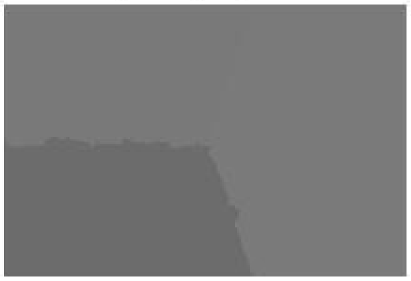
EM: 3 segments (Grayscale & Position)



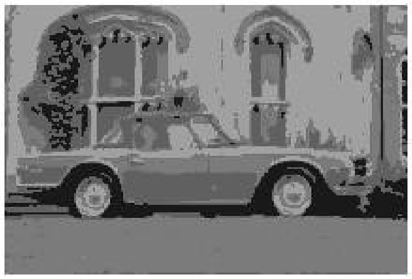
k-means: 3 segments (Grayscale)



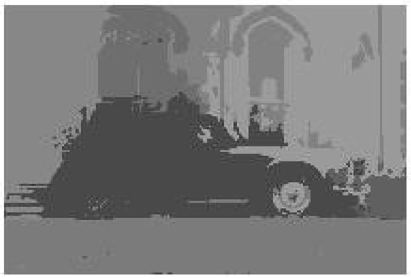
k-means: 3 segments (Grayscale & Position)



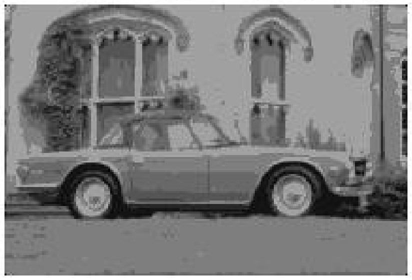
EM: 7 segments (Grayscale)



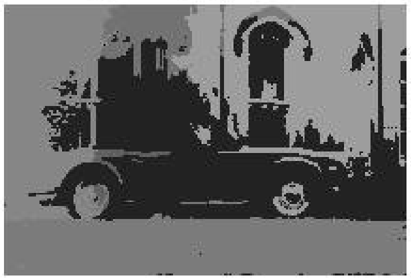
EM: 7 segments (Grayscale & Position)



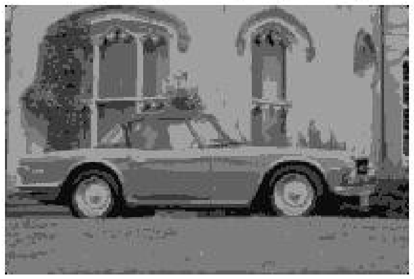
k-means: 7 segments (Grayscale)



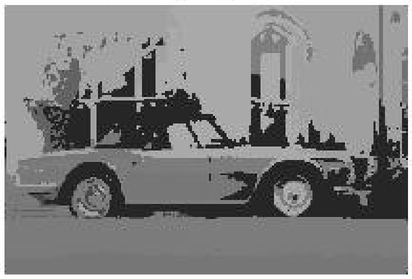
EM: 7 segments (Grayscale & Position)



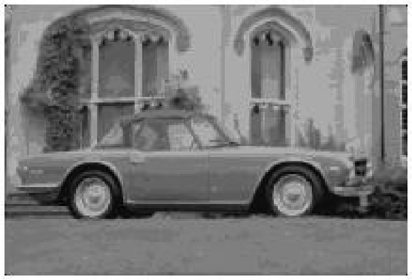
EM: 13 segments (Grayscale)



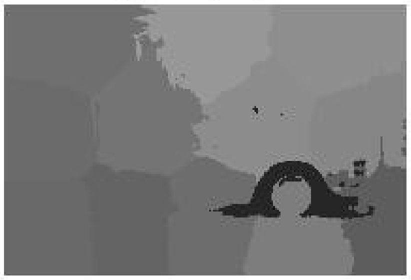
EM: 13 segments (Grayscale & Position)

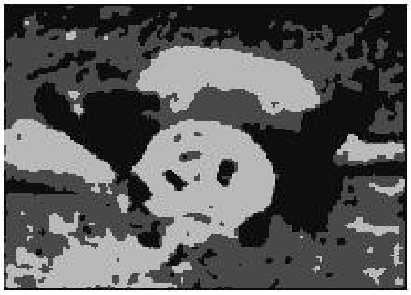


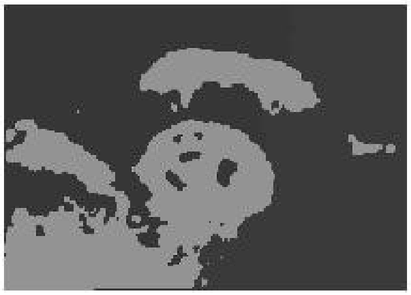
k-means: 13 segments (Grayscale)

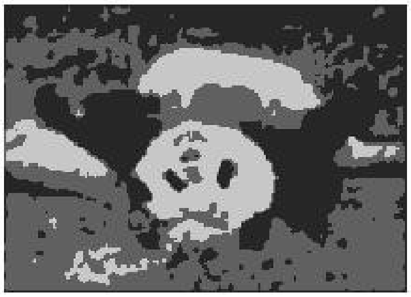


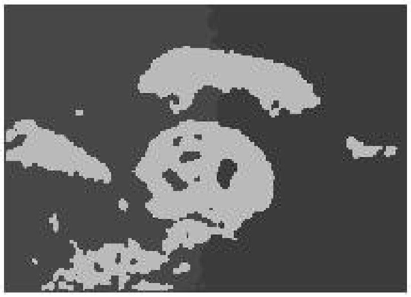
k-means: 13 segments (Grayscale & Position)

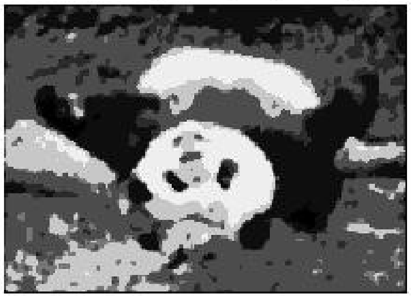


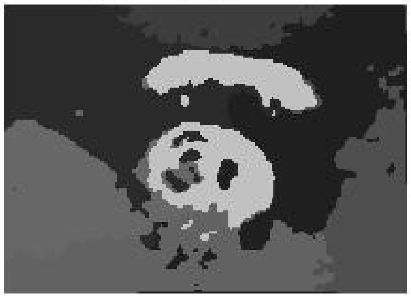

EM: 3 segments (Grayscale)

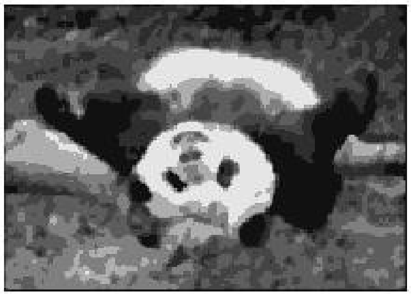

EM: 3 segments (Grayscale & Position)

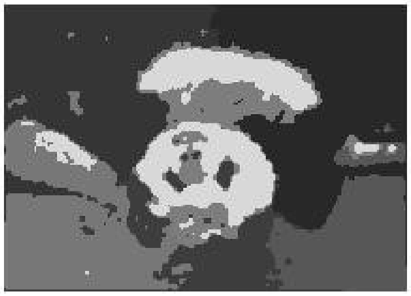

k-means: 3 segments (Grayscale)

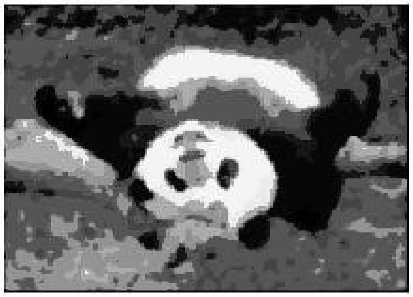

k-means: 3 segments (Grayscale & Position)

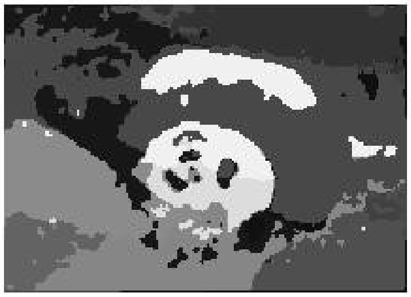

EM: 7 segments (Grayscale)


EM: 7 segments (Grayscale & Position)

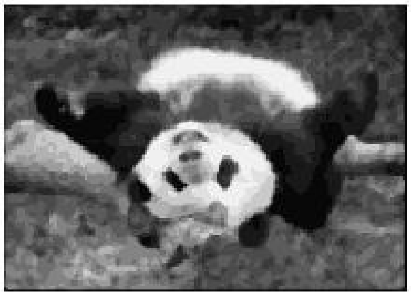

k-means: 7 segments (Grayscale)


k-means: 7 segments (Grayscale & Position)

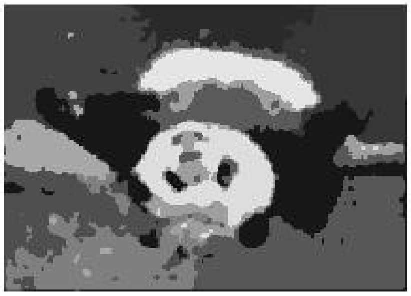

EM: 13 segments (Grayscale)


EM: 13 segments (Grayscale & Position)

k-means: 13 segments (Grayscale)



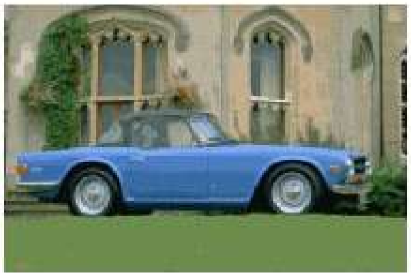
k-means: 13 segments (Grayscale & Position)



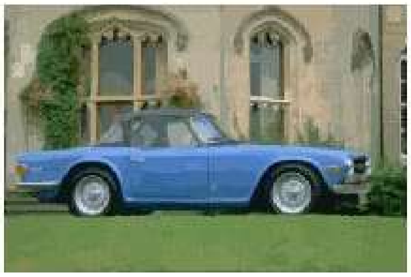
The choice of using grayscale info itself already makes results worse as color info are totally lost and pixels that belong to different items are classified into one cluster even though position info us not used.
For grayscale images, the tendency of better result coming with more cluster still does not change, and KMeans with only grayscale info resembles best a grayscale version of original picture. However, position info hinders more seriously on segmentation result and give far worse result in comparison to only using grayscale features.

3.2

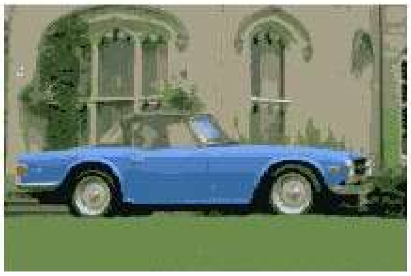
Mean-Shift: window size=7 (RGB)



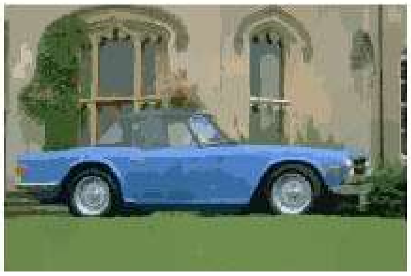
Mean-Shift: window size=7 (RGB & Position)



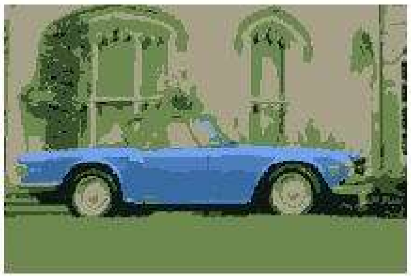
Mean-Shift: window size=25 (RGB)



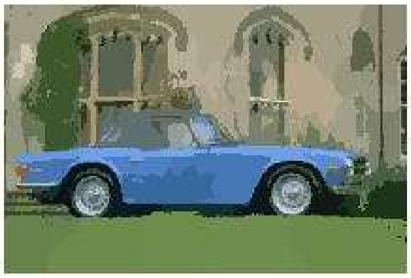
Mean-Shift: window size=25 (RGB & Position)



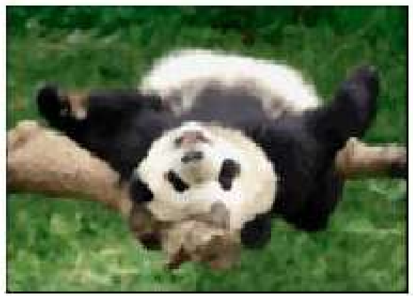
Mean-Shift: window size=37 (RGB)



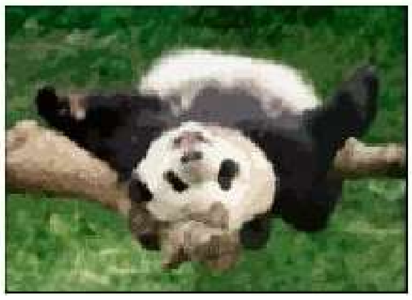
Mean-Shift: window size=37 (RGB & Position)



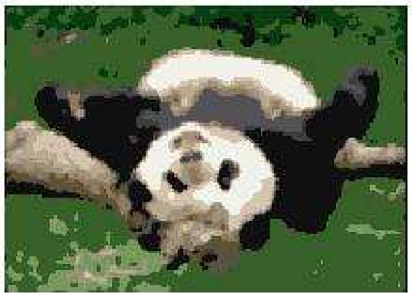
Mean-Shift: window size=7 (RGB)



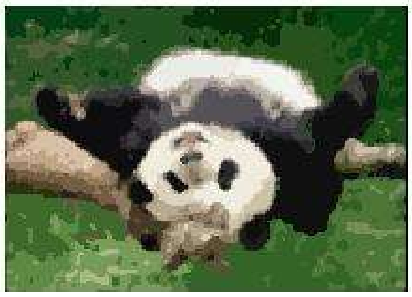
Mean-Shift: window size=7 (RGB & Position)



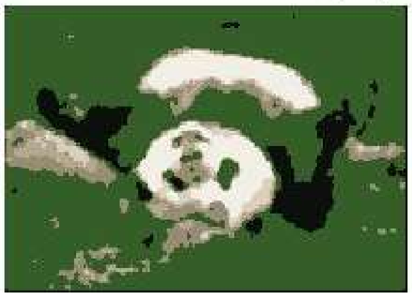
Mean-Shift: window size=25 (RGB)



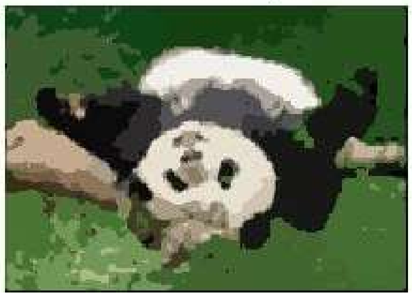
Mean-Shift: window size=25 (RGB & Position)



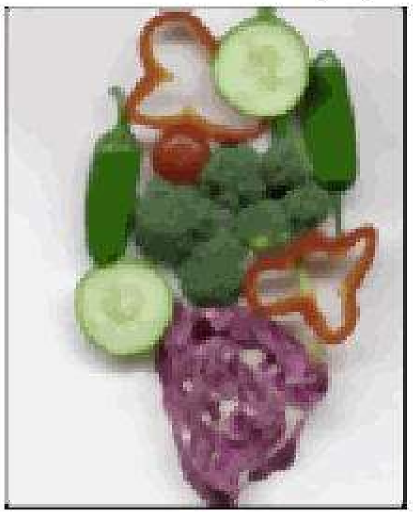
Mean-Shift: window size=37 (RGB)



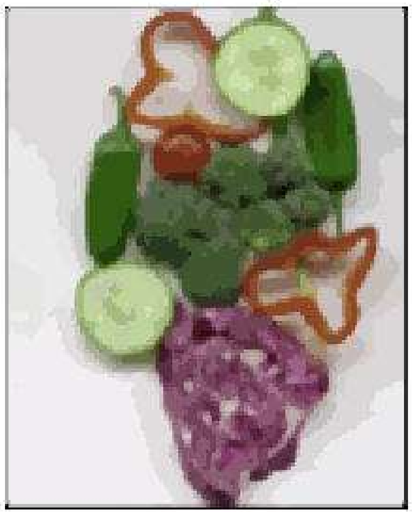
Mean-Shift: window size=37 (RGB & Position)



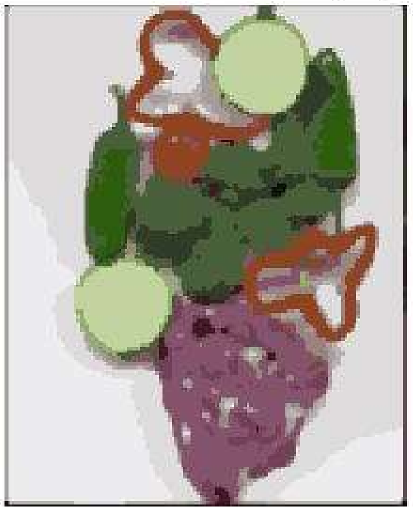
Mean-Shift: window size=7 (RGB)



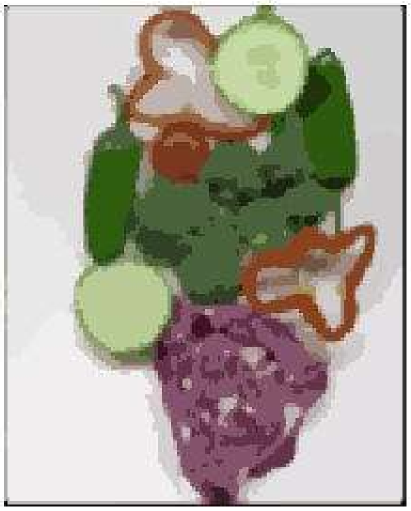
Mean-Shift: window size=7 (RGB & Position)



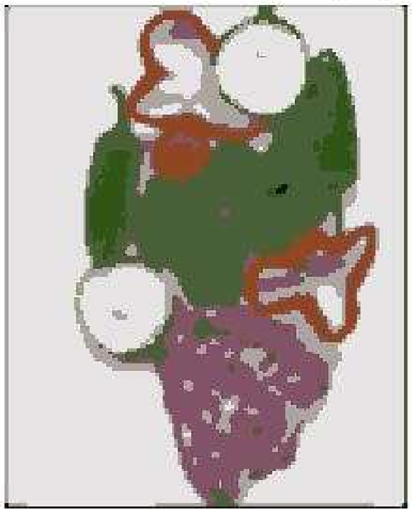
Mean-Shift: window size=25 (RGB)



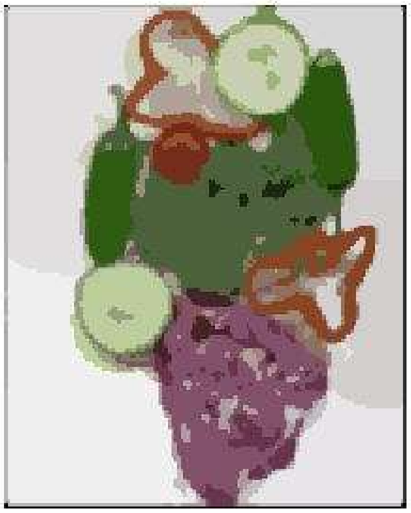
Mean-Shift: window size=25 (RGB & Position)



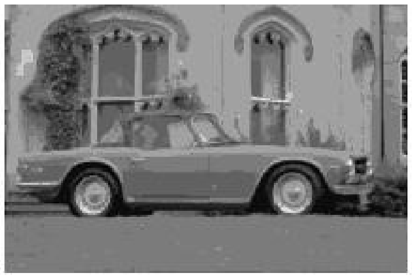
Mean-Shift: window size=37 (RGB)



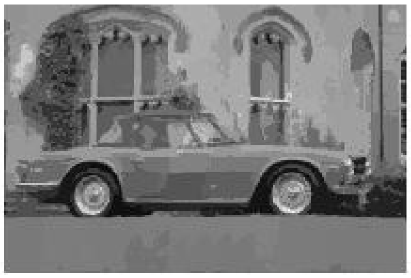
Mean-Shift: window size=37 (RGB & Position)



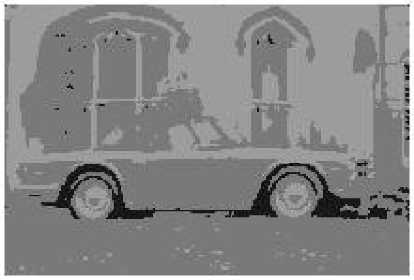
Mean-Shift: window size=7 (Grayscale)



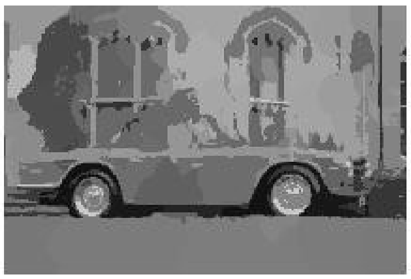
Mean-Shift: window size=7 (Grayscale & Position)

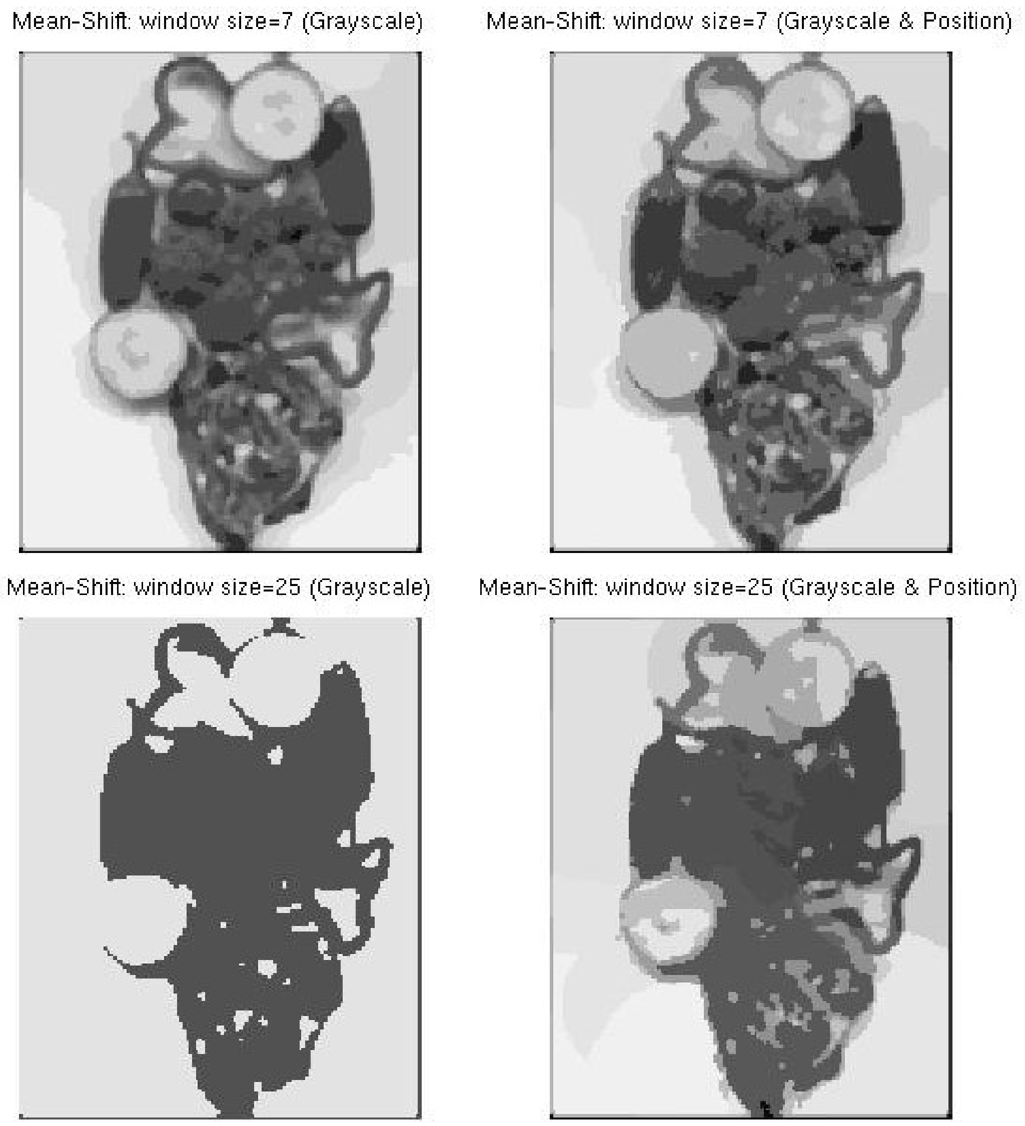


Mean-Shift: window size=25 (Grayscale)



Mean-Shift: window size=25 (Grayscale & Position)




Mean-Shift: window size=7 (Grayscale)
Mean-Shift: window size=7 (Grayscale & Position)
Mean-Shift: window size=25 (Grayscale)
Mean-Shift: window size=25 (Grayscale & Position)

Generally speaking, smaller the window size, better the result. For colored image, compared to using RGB feature alone, combining it with position info will not do obvious hinderance to results in mean shift clustering; rather, it gives better result when window size is large. For gray scale image, same size of window will give worse segmentation results regarding both detail and segmentation accuracy due to loss of color info.