Lab 07 - BSDS

Jhony A. Mejia

Universidad de Los Andes

Biomedical Engineering Department

ja.mejia12@uniandes.edu.co

Abstract

Clustering can be used for segmentation and contour detection. However, clustering performs better in the first task than in the second one. The most common clustering methods are kmeans, gmm, hierarchical segmentation and watershed. Kmeans in general performs better when one doesn't know what to do. Anyway, the clustering method result is highly affected by the feature representation of the image. In this case, the best feature representation was obtained with RGB (color) and X and Y (spatial). Also it is important to recall that downsampling an image can be useful for making faster methods without diminishing the method's performance (AP or Jaccard). Finally, post-segmentation processes can be very useful for improving a method's performance.

1. Introduction

There are different methods for clustering data. One of the most used methods is k-means. K-means is a process of optimization of groups. The idea is to make that the data belonging to a cluster is more similar with data within the cluster than outside it. The most commonly distance used in k-means is the euclidean distance. The clusters are 'hard assignments' (which means, each cluster has a label). In general, k-means is the easiest clustering method. Additionally, k-means performs in general better than any other clustering algorithm.

Clustering can be used for producing segmentations. The features that are presented to the methods have a wide variety. One of the most common features are color (and its different color spaces representation) and space information (distance between points). For obtaining an optimal segmentation one must find the best combination of features, method and its hyper-parameters.

BSDS500 is one of the most important databases for studying segmentation/border detection problems. It contains 500 images (200 for train, 200 for test and 100 for validation). Each image has multiple annotations made by humans. The goal is to produce an algorithm that resembles human's performance in both border detection and segmentation.

2. Materials and Methods

2.1. Baseline from previous works

In previous works [1] it was showed that the best methods for producing segmentations were obtained with k-means. The best result was obtained with k-means and HSV color space, and the second best result was obtained with k-means and RGB+XY. The 'best method' of the previous work was defined taking into account the relationship between time and Jaccard's index. The performance of this methods of previous works can be seen in Table 1

Feat Space	Clustering Method	ACA
HSV	k-means	33.84%
RGB+XY	k-means	33.89%

Table 1. Best two methods obtained from previous works with its respective ACA.

2.2. Baseline tested in a BSDS mini-database

For reproducing and reconfirming the two best method in BSDS, three methods were tested in a mini-database (bench-fast images, containing 5 images). The methods compared in this data base were k-means in HSV, k-means in RGB+XY, and gmm in Lab. For each method, qualitative and time performance was recorded. The clusters taken into account for this process were 5, 10, 15 and 20.

After that, it was confirmed that the best two methods were the ones found in the previous work (HSV-kmeans and RGB+XY-kmeans). It is important to say that none rescaling was performed in any of the feature spaces. Additionally, the methods were ran with the default parameters (excepting gmm, where 'Regularization Value' was settled to 0.1).

2.3. Baseline's testing in BSDS

20 segmentations were done per images. The k-clusters started in 5, and augmented by 5 until 100 clusters, and each k produced a segmentation. The two default (raw) best methods were tested in the test subset of BSDS500 and the results obtained were compared with Pablo Arbelaez' method [2].

2.4. Baseline's hyper-parameters for improvement

More hyper-parameters were added to the model in order to improve its performance. Two resolution reduction methods were used (Gaussian downsampling (G) and cubic interpolation (I)). Two levels of reduction were considered (half (1) or quarter (2) of the original image). These two hyper-parameters were related to the image itself. Two additional hyper-parameters were defined for the k-means algorithm. Those were the maximum of iterations (500 or 1000) and centroids initializing ('plus' or 'cluster'). These hyper-parameters were tested in the BSDS500's validation subset (100 images).

2.5. Baseline improved evaluation

The two best methods after considering these hyper-parameters were tested in the test subset (200 images). These two methods were also compared with Pablo Arbelaez' method [2].

The metrics analyzed for all the methods tested were the Precision-Recall curves, AP, Jaccard Index and relationship between OIS and ODS.

3. Results

3.1. Time response in a mini-database of three methods used in previous works

As stated before, gmm in Lab, k-means in HSV and k-means in RGB+XY were performed in a mini-database of 5 images for obtaining an approximation of the method's performance (Table 3).

Feat Space	Cluster Method	time	time/image
HSV	k-means	171sec	34.1sec
RGB+XY	k-means	221sec	44.3sec
Lab	gmm	2159sec	431.8sec

Table 2. Time response of three methods that were shown to have the best ACA's in previous works. This time response was obtained in a sub-database of 5 images (bench_fast) with 4 segmentations per image (k=5, 10, 15 and 20)

Taking into account the time factor, the best method is HSV-kmeans, followed by RGB+XY-kmeans, and at last Lab-gmm. Qualitative results can be seen in Figures 1 - 6.

3.2. Qualitative results in a mini-database of three methods used in previous works

As one can see in Fig. 1, HSV - kmeans makes a decent perceptual segmentation. It's most notable mistake is the shadow of the tables. The mountains could be segmented by another person. In Fig. 2 it is easy to detect that this method has a really high recall (almost all of the segmented borders by the person are present in the predicted borders) but a really low precision (too many false positives).
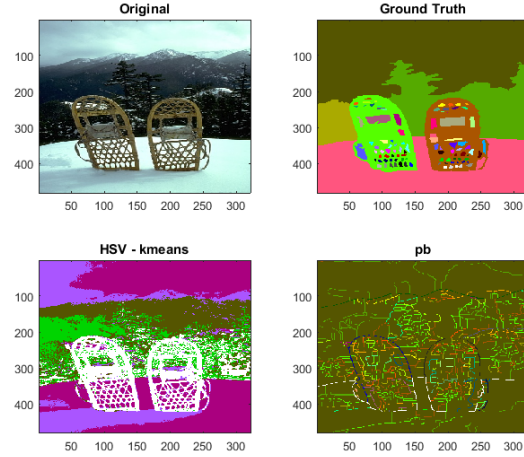

Figure 1. Comparison of segmentations between the original image, a given segmentation by a person, HSV - kmeans segmentation and Pablo's method (Probability of Boundary - Pb).
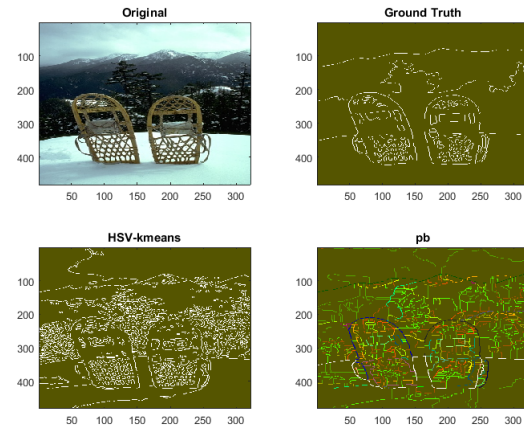

Figure 2. Comparison of border detection between the original image, a given segmentation by a person, HSV - kmeans segmentation and Pablo's method (Probability of Boundary - Pb).

In Fig. 3 it is easy to recognize the debilities of RGB+XY - kmeans in segmentation. This method tends

to divide homogeneous regions because they are far away. However, in 4 one can see that these method has a higher precision than HSV - kmeans, but a lower recall, despite of the creation of false contours (as in the middle of the image).
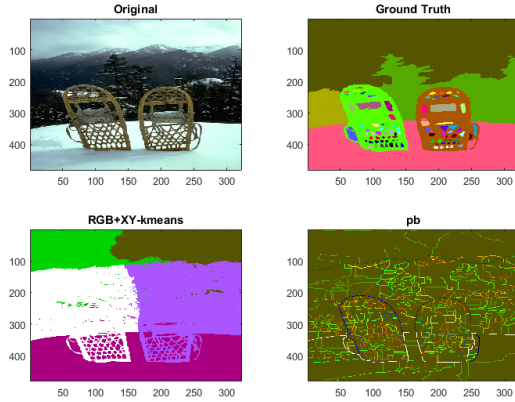

Figure 3. Comparison of segmentations between the original image, a given segmentation by a person, RGB+XY - kmeans segmentation and Pablo's method (Probability of Boundary - Pb).
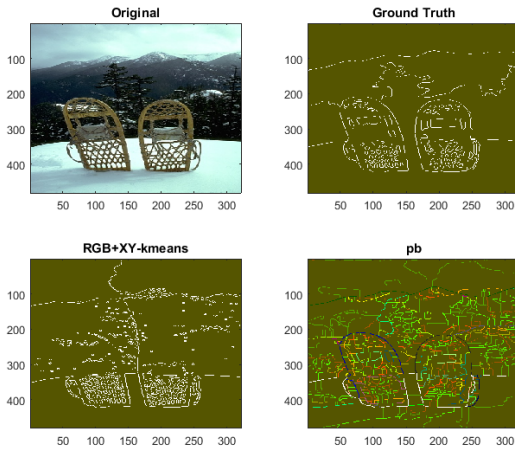

Figure 4. Comparison of border detection between the original image, a given segmentation by a person, RGB+XY - kmeans segmentation and Pablo's method (Probability of Boundary - Pb).

In Fig. 5 it can be seen that Lab - gmm method produces acceptable segmentations. Anyway, the presence of false segmentations is easy to recognize. It's more common mistake is seen in bright locations (as the clouds in the sky or the snow that is directly being impacted by the sun). In Fig. 6 the result of contour detection is similar to the one obtained in HSV - kmeans. However, this method produces

a bit more of false positives (like in the lower right part) and has almost the same recall than HSV - kmeans.
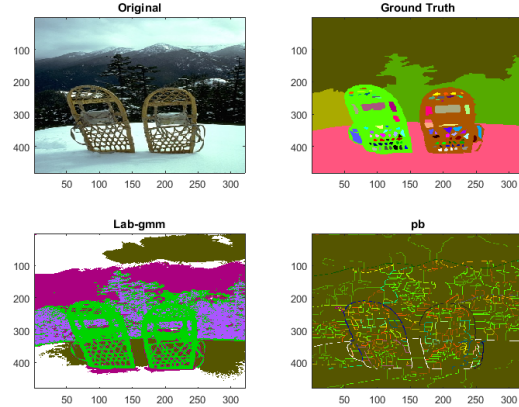

Figure 5. Comparison of segmentations between the original image, a given segmentation by a person, Lab - gmm segmentation and Pablo's method (Probability of Boundary - Pb).
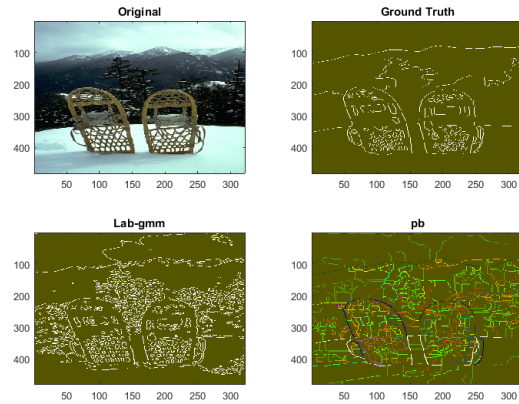

Figure 6. Comparison of border detection between the original image, a given segmentation by a person, Lab - gmm segmentation and Pablo's method (Probability of Boundary - Pb).

To sum up, the most noticeable patterns of mistakes of HSV - kmeans and Lab - gmm are the creation of false contours. This false contours can be created mainly by shadows or changes in luminosity.

3.3. Definition of the two best methods

Taking into account that the qualitative results of HSV - kmeans compared with Lab - gmm, and considering that Lab - gmm takes 7 minutes per image while HSV - kmeans only takes 34 seconds per image; one of the selected as 'best methods' is HSV - kmeans. It is important to say that this was the best method in the previous work [1].

3

Similarly, RGB+XY will be considered as the 'second best method' because of its qualitative high precision.

3.4. Evaluation in BSDS500's test subset of the two best methods

The two best methods were tested in BSDS500's test subset. First, the segmentations were created using 'runMethod.m', which contains several functions from previous works [1]. After that, the Precision-Recall curve was created using 'evalData.m' and 'plot_eval'. The Precision-Recall curves obtained can be seen in 7.
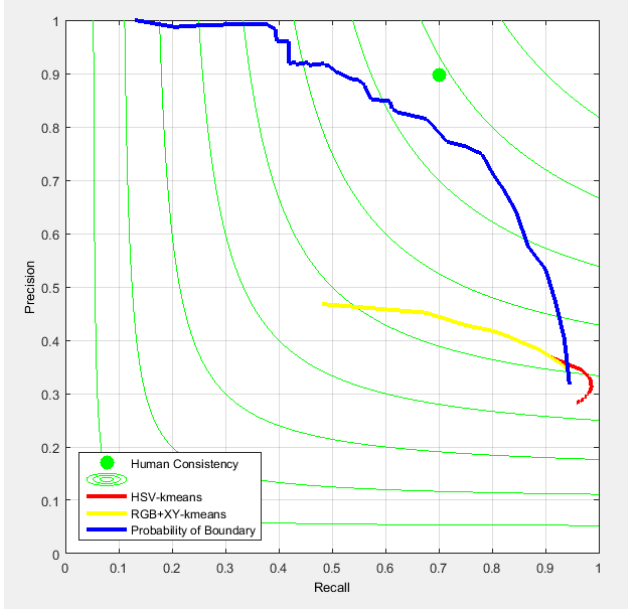

Figure 7. Precision-Recall curves for the proposed methods (HSV - kmeans and RGB+XY - kmeans) and Probability of Boundaries (Pablo's method).

It is important to say that ideally, one should have a Precision - Recall curve that is always (or at least most of the time) in (1,1). However, it would be weird that the algorithm produces borders and segmentations that are even better than the ones of humans. Taking that into account, the practical goal is to be near of the human consistency (their F-Value).

The 'runMethod.m' times (creation of segmentations) in the test subset are:

Feat Space	Cluster Method	time	time/image
HSV	k-means	143387sec	12min
RGB+XY	k-means	209093sec	17min

Table 3. Time response of the two best methods in BSDS's test subset. 20 segmentations per image (k=5:5:100)

One can see that the time performance of both methods is

extremely poor. More than 10 minutes per images is really slow. Also, it is important to say that both methods showed a lot of failures of convergence. RGB+XY showed significant more convergence failures than HSV. It may suggest that the clustering process might not be optimal.

3.5. Best method's performance analysis and comparison with Arbelaez Pablo

As one can see in 7, the method of HSV - kmeans has a really poor performance. It only has values of high recall and very low variations on precision. It even does a strange U-turn in some part (unknown behavior). It's highest precision does not go over 0.4. More detailed information of this method's performance can be seen in Fig. 8. There, one can see that this method's performance as contour detector is really poor (AP of 0.02). However, the method's performance as a segmentation algorithm is acceptable (Jaccard of 0.41). It is important to say that despite of having a poor performance, this method is robust as the OIS and ODS are similar. The best results are obtained in k = 5 (it says th=1, but the 1 corresponds to k=5).

```
Eval_HK
Boundary
ODS: F( 0.91, 0.37 ) = 0.52    [th = 1.00]
OIS: F( 0.94, 0.38 ) = 0.54
Area_PR = 0.02

Region
GT covering: ODS = 0.38 [th = 1.00]. OIS = 0.38. Best = 0.41
Rand Index: ODS = 0.73 [th = 2.00]. OIS = 0.76.
Var. Info.: ODS = 2.99 [th = 1.00]. OIS = 2.98.
```

Figure 8. Detailed performance results of HSV - kmeans method.

The method of RGB+XY - kmeans (Fig. 7) shows a more dynamical performance. It has a wider range of precisions and recalls. However, the maximum recall is slower than most of the HSV - kmeans method. Anyway, this method has a better performance as contour detector (AP = 0.20) as it can be seen in Fig. 9. Also, this method performance as a segmentation algorithm is slightly better (Jaccard of 0.46). Similarly to HSV, this method has similar OIS and ODS, which suggests robustness. The best results are obtained in k = 20 for contour detection and k = 5 for segmentation.

```
Eval_RXK
Boundary
ODS: F( 0.80, 0.42 ) = 0.55    [th = 4.00]
OIS: F( 0.84, 0.45 ) = 0.58
Area_PR = 0.20

Region
GT covering: ODS = 0.35 [th = 1.00]. OIS = 0.37. Best = 0.46
Rand Index: ODS = 0.74 [th = 2.00]. OIS = 0.75.
Var. Info.: ODS = 2.67 [th = 1.00]. OIS = 2.63.
```

Figure 9. Detailed performance results of RGB+XY - kmeans method.

Both methods are hugely overwhelmed by Pablo Arbelaez' method [2]. As one can see in Fig. 7, the curve is almost always above both of the methods proposed. The only part that seems to 'beat' Pablo is that HSV - kmeans has a greater recall (right part of the graph). However, in the balance between precision and recall, Pablo beats us with a huge margin. As one can see in 10, his method performance is remarkable both as a contour detector (AP = 0.68) and as a segmentation algorithm (Jaccard of 0.84). Also, his method is robust as it shows similar OIS and ODS.

```
C:\Users\officedepot\Documents\MATLAB\Lab7\BSR\bench_fast\eval\test_all_fast
Boundary
ODS: F( 0.78, 0.75 ) = 0.76    [th = 0.10]
OIS: F( 0.78, 0.81 ) = 0.80
Area_PR = 0.68

Region
GT covering: ODS = 0.66 [th = 0.23]. OIS = 0.77. Best = 0.84
Rand Index: ODS = 0.83 [th = 0.14]. OIS = 0.92.
Var. Info.: ODS = 1.36 [th = 0.28]. OIS = 1.03.
```

Figure 10. Detailed performance results of Probabilty of Boundary (Pablo's method).

It is fair to say that the author of this paper does not know how long does Pablo's method takes. However, Pablo's algorithm does not lose time using k-means or any other clustering algorithm. Pablo has a huge advantage, and it is that he is taking into account texture, while we are only taking into account intensity. It is clever to think that texture can be more effective for detecting contours than only changes in intensity. To sum up, the process of representing the image in a specific feature space (texture in Pablo's method and intensity and/or location in our method) is more effective in Pablo's method, but more expensive.

Pablo's method takes into account multiple scale analysis. This can offer him a wider range of borders. We only consider one scale at a time for simplicity. Again, this part might be slower in Pablo's algorithm compared with ours. However, the richness of Pablo's contours is way bigger than ours. Also Pablo uses the local differences of textures using Probability of Boundaries. Anyway, one might think that Pablo's algorithm might have a great recall, but a lot of false positives, as our HSV - kmeans method.

Pablo solves the problem of the false positives by using normalized cuts with its eigenvectors. Again, he doesn't use

k-means. Instead he uses the gradient of the eigenvectors. After using this, Pablo reduces and prevents drastically the number of false positives. We don't use any algorithm for reducing or preventing false positives.

Pablo's algorithm is way more refined that ours. He takes multiple steps for optimizing his results. He never uses k-means, as it takes too long to process. This makes his algorithm way more effective. As Pablo himself says, when you don't know what to do, you use kmeans. He does know what to do for improving results. Meanwhile, our segmentation and contour detection algorithm is based in k-means. However, k-means produces good results, compared with its simplicity.

3.6. Two best methods improvement

After being aware that rising the AP and the Jaccard index is really difficult only using k-means, our efforts went on developing a faster method that showed similar quality performance.

3.6.1 Image's dimensionality

Two hyper-parameters were defined for improving the image representation. The first hyper-parameter is associated with the modality of reducing resolution. Two modalities were considered: Gaussian reduction and bicubic interpolation. The second hyper-parameter is related to the level of reduction. Two levels were considered: half or quarter of the original resolution. The idea of reducing the resolution is making faster the process of clustering (segmentation).

One of the main challenges of reducing dimensionality is how to compare it with it's corresponding ground truth. Basically there are two options: lower the annotation's resolution or returning the segmentation's resolution to the original image dimensions. The first option results on segmentations and boundaries can be seen in Fig. 11 and 12. As it can be seen, the effect on reducing the dimensionality of the segmentation using Gaussian scaling produces new unwanted regions. The effect on the borders is devastating: it destroys almost all borders.
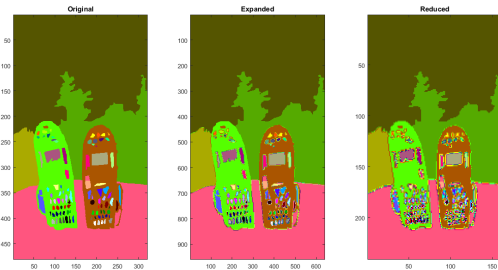

Figure 11. Gaussian effect on segmentation after upsampling or downsampling.
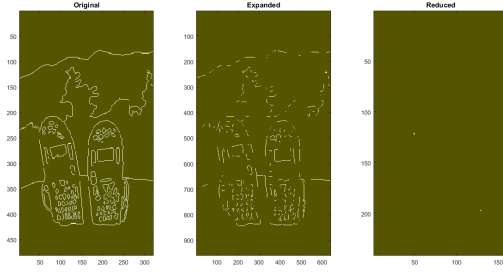
Figure 12. Gaussian effect on borders after upsampling or down-sampling.

That is why the second option had to be considered. Again, two options for upsampling are considered (gaussian and interpolation). The upsampling of the segmentation can be seen in Fig. 13. Again it is easy to see that new regions are being created when using Gauss, while almost the same image is obtained using Interpolation with its nearest neighbor. Some destruction of borders can be seen in Gaussian upsample, while borders are conserved using interpolation with the nearest neighbor (Fig. 14). That is why interpolation with nearest neighbor was chosen as the method for upsampling and posterior comparison with ground truth.
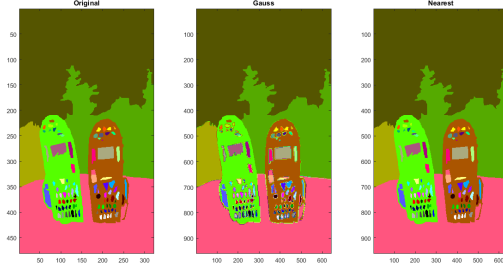

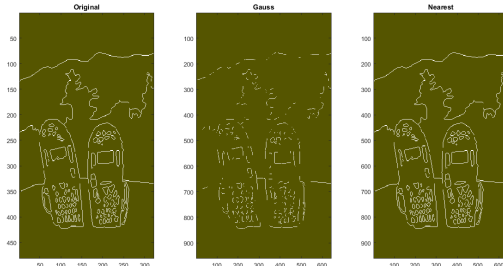Figure 13. Segmentation upsampling using Gaussian or Interpolating method.

Figure 14. Border upsampling using Gaussian or Interpolating method.

To sum up, to options were considered to downsample

the original image for its posterior segmentation (Gauss or Bicubic Interpolation). For upsampling (for comparing with ground truth), interpolation by nearest neighbor was performed. This process was repeated depending on the reducing level considered (1 or 2).

3.6.2 Clustering method

Two other hyper-parameters were considered for improving the clustering process. These two parameters were the maximum of iterations and the initialization of centroids. Both hyper-parameters' goal is to guarantee that the model will converge.

3.7. Evaluation of hyper-parameters for improvement in validation subset

The new method with 4 hyper-parameters was tested in the val subset. Each method took the following times:

Cluster. Method	Max Iter.	Cluster init.	Red. Method	total time	time / image
HSV-k	500	plus	Gaussian	15h	9min
RGB+XY-k	500	plus	Gaussian	16h	10min
HSV-k	500	plus	Interpol	15h	9min
RGB+XY-k	500	plus	Interpol	16h	10min
HSV-k	500	cluster	Gaussian	19h	11min
RGB+XY-k	500	cluster	Gaussian	17h	10min
HSV-k	500	cluster	Interpol	19h	11min
HSV-k	500	cluster	Interpol	18h	11min

Table 4. Time response of the hyper-parameters of the two best methods in BSDS's validation subset. The only hyper-parameter fixed is the level of reduction (half the size). 20 segmentations per image (k=5:5:100).

Cluster. Method	Max Iter.	Cluster init.	Red. Method	total time	time / image
HSV-k	500	plus	Gaussian	2h	1min
RGB+XY-k	500	plus	Gaussian	2h	1min
HSV-k	500	plus	Interpol	2h	1min
RGB+XY-k	500	plus	Interpol	2h	1min
HSV-k	500	cluster	Gaussian	2h	1min
RGB+XY-k	500	cluster	Gaussian	3h	2min
HSV-k	500	cluster	Interpol	3h	2min
HSV-k	500	cluster	Interpol	3h	2min

Table 5. Time response of the hyper-parameters of the two best methods in BSDS's validation subset. The only hyper-parameter fixed is the level of reduction (quarter the size). 20 segmentations per image (k=5:5:100).

The objective of reducing time was accomplished with the hyper-parameters purposed. It's corresponding

Precision-Recall curves can be seen in Figs. 15 and 16. As it can be seen, the best methods for both levels of downsampling (1 and 2) are RGB+XY with Gaussian downsampling and its corresponding level, max of iterations and any of the clustering method initialization.
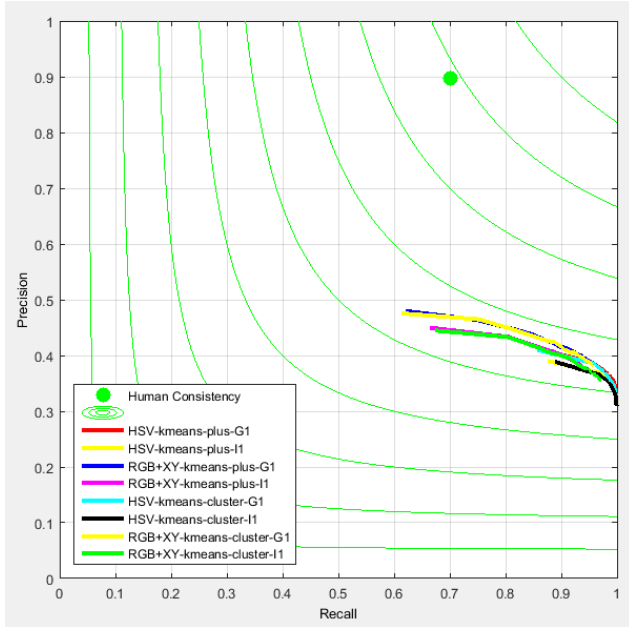

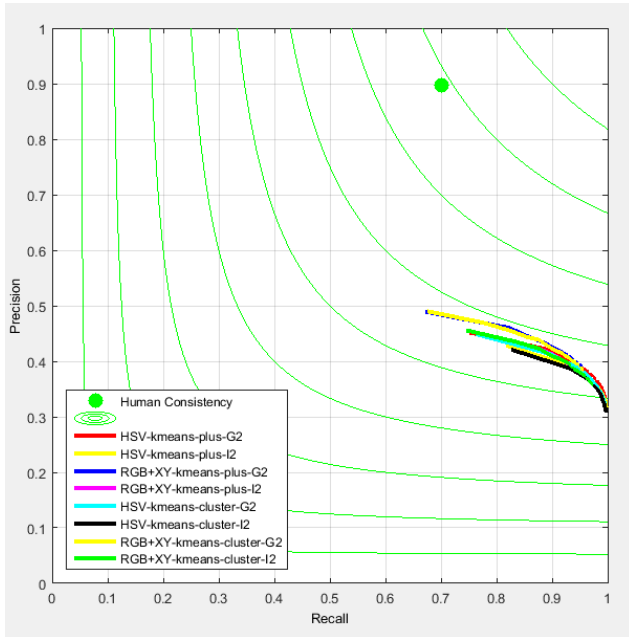Figure 15. Validation Precision-Recall curves for different hyper-parameter's values (1 level of downsample).

Figure 16. Validation Precision-Recall curves for different hyper-parameter's values (2 levels of downsample).

3.7.1 Best two methods after hyper-parameters' performance

The specific results for the best methods can be seen in Figs. 17, 18, 19 and 20. There was no difference on the clustering initialization method, but it is fair to say that the 'cluster' method was slower than the 'plus' method. Little difference was obtained in the results of 1 level of downsampling, compared with the ones of 2 levels (AP=0.15, J=0.45 and AP=0.13, J=0.44 respectively).

```
Eval_RXK_cluster_G_1
Boundary
ODS: F( 0.81, 0.45 ) = 0.58    [th = 3.00]
OIS: F( 0.85, 0.47 ) = 0.61
Area_PR = 0.15

Region
GT covering: ODS = 0.37 [th = 1.00]. OIS = 0.38. Best = 0.45
Rand Index: ODS = 0.72 [th = 2.00]. OIS = 0.74.
Var. Info.: ODS = 2.68 [th = 1.00]. OIS = 2.64.
```

Figure 17. Validation results for RGB+XY, 500 maximum iterations, 'cluster' clustering, Gaussian downsampling (1 level of downsample).

```
Eval_RXK_plus_G_1
Boundary
ODS: F( 0.84, 0.44 ) = 0.58    [th = 3.74]
OIS: F( 0.85, 0.47 ) = 0.61
Area_PR = 0.15

Region
GT covering: ODS = 0.37 [th = 1.00]. OIS = 0.38. Best = 0.45
Rand Index: ODS = 0.72 [th = 2.00]. OIS = 0.74.
Var. Info.: ODS = 2.66 [th = 1.00]. OIS = 2.63.
```

Figure 18. Validation results for RGB+XY, 1000 maximum iterations, 'plus' clustering, Gaussian downsampling (1 level of downsample).

```
Eval_RXK_cluster_G_2
Boundary
ODS: F( 0.82, 0.46 ) = 0.59    [th = 3.00]
OIS: F( 0.86, 0.48 ) = 0.61
Area_PR = 0.13

Region
GT covering: ODS = 0.36 [th = 1.00]. OIS = 0.38. Best = 0.44
Rand Index: ODS = 0.72 [th = 2.00]. OIS = 0.74.
Var. Info.: ODS = 2.86 [th = 1.00]. OIS = 2.82.
```

Figure 19. Validation results for RGB+XY, 500 maximum iterations, 'cluster' clustering, Gaussian downsampling (2 levels of downsample).

```
Eval_RXK_plus_G_2
Boundary
ODS: F( 0.82, 0.46 ) = 0.59   [th = 3.00]
OIS: F( 0.86, 0.48 ) = 0.62
Area_PR = 0.13

Region
GT covering: ODS = 0.36 [th = 1.00]. OIS = 0.38. Best = 0.44
Rand Index: ODS = 0.72 [th = 2.00]. OIS = 0.74.
Var. Info.: ODS = 2.85 [th = 1.00]. OIS = 2.82.
```

Figure 20. Validation results for RGB+XY, 1000 maximum iterations, 'plus' clustering, Gaussian downsampling (2 levels of downsample).

Considering time and efficiency, the best two methods for the new hyper-parameters are 2-RGB+XY-G-Plus-1000 and 1-RGB+XY-G-Plus-500. Both methods were ran in the test method to compare the performance after the hyper-parameter optimization.

3.8. Evaluation of the two best methods with optimal hyper-parameters in the test set

The performance of the best methods can be seen in Table 6 and Figs. 21, 22 and 23.

Reduction level	Max iter.	total time	time / image
2	1000	2h	28sec
1	500	8h	2.5min

Table 6. Time response of the two best methods with optimal hyper-parameters in BSDS's test subset. 20 segmentations per image (k=5:5:100).
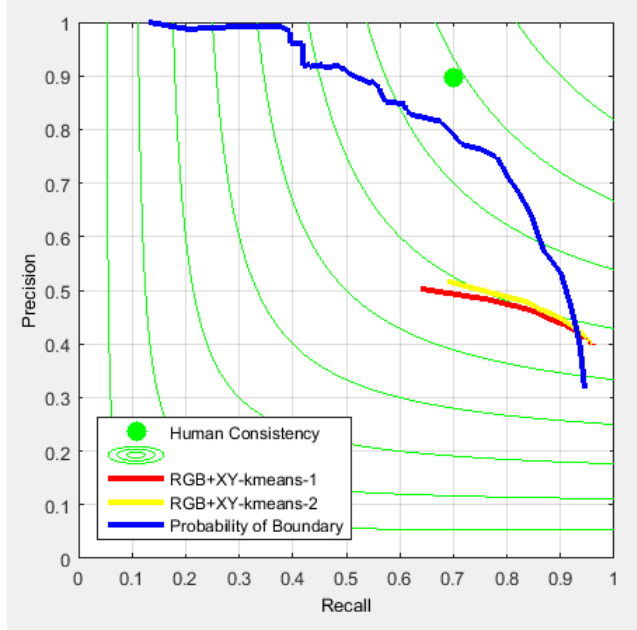

Figure 21. Precision Recall curves for comparing the best two methods with optimal hyper-parameters with Pablo Arbelaez' method.

```
Eval_fRXK1
Warning: MATLAB has disabled some advanced graphics rendering
information, click here.
Boundary
ODS: F( 0.85, 0.46 ) = 0.60   [th = 3.72]
OIS: F( 0.86, 0.48 ) = 0.62
Area_PR = 0.15

Region
GT covering: ODS = 0.37 [th = 1.00]. OIS = 0.39. Best = 0.47
Rand Index: ODS = 0.74 [th = 2.00]. OIS = 0.76.
Var. Info.: ODS = 2.73 [th = 1.00]. OIS = 2.67.
```

Figure 22. Detailed performance of the best method after hyper-parameter optimization (1 level reduction and 500 maximum iterations with Gaussian reduction). Optimal k is equal approximately to 19.

```
Eval_fRXK2
Boundary
ODS: F( 0.84, 0.48 ) = 0.61   [th = 3.00]
OIS: F( 0.85, 0.50 ) = 0.63
Area_PR = 0.13

Region
GT covering: ODS = 0.37 [th = 1.00]. OIS = 0.39. Best = 0.46
Rand Index: ODS = 0.74 [th = 3.00]. OIS = 0.76.
Var. Info.: ODS = 2.87 [th = 1.00]. OIS = 2.82.
```

Figure 23. Detailed performance of the second best method (2 levels reduction and 1000 maximum iterations with Gaussian reduction). Optimal k is equal to 15.

The results obtained were practically the same in the two

methods. A slight improvement is seen in the best method (1 level of reduction and 500 maximum iterations). However, the excess of time that it took compared with the second best method was not worth it. Anyway, both methods are way faster than the original two methods.

The performance of the original results is really similar to the performance of the improved methods. The difference is mainly seen in the AP, which was 0.2 for the original RGB+XY - kmeans method, and is 0.15 and 0.13 for the two best methods after optimization respectively. This means that the performance as a contour detector is better in the original method.

The performance as a segmentation algorithm is the same in the original and the optimized methods. To sum up, if the user wants to segment images, the improved version of the algorithm is suggested. If the user wants a contour detector, he/she must decide between a slight improvement in ACA (0.2 in the original and 0.15 in the best improved method) and time per image (17mins in the original and 2.5mins in the best improved method).

In general the goal of reducing the time of the algorithm without reducing significantly its performance was achieved.

Qualitative performance of the best improved method

The qualitative performance of a given image with a human segmentation can be seen in Fig. 24.
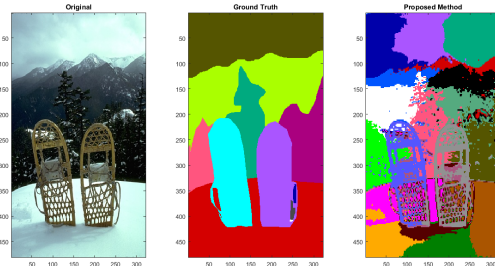

Figure 24. Comparison between the original image (left), ground truth by a person (center) and the proposed method (right) of RGB+XY and k-means with 1 level of reduction using Gaussian downsampling, 19 clusters, 'plus' cluster initiation, and 500 maximum iterations.

As it can be seen, the algorithm roughly takes into account the same regions that the person took. The problem is that the algorithm tends to over segment the image considering spatial information. That means that some regions like the top of the image is divided in three, when it is clear that the sky should be considered as only one entity.

The algorithm also tends to make mistakes in shadows like in the bottom of the picture (the floor). The algorithm considers that the shadow itself is another entity, when it is clear that the floor is only one thing. To sum up, this method makes mistakes both in intensity (shadows) and far away components (spatial information).

However the algorithm performance is decent. As it was shown in the numerical performance, the segmentation behavior is good (almost 50%). The other 50% remaining can be obtained with a method for reducing sub segmentation as normalized cuts. This should solve the problem of the spatial partitioning and intensity issues that RGB+XY tends to make. It is important to recall that most of the original segmentations can be obtained by merging sections, which gives hope that this method could be useful after some post-processing work.

3.9. General analysis and further improvements

Limitations and possible solutions to it

One of the biggest limitation of the proposed algorithms is the complexity for defining the optimal parameters. Having to change the resolution, the method for changing dimensions, and k-means hyper-parameter is exhausting. It is even more exhausting when one doesn't see significant differences between methods. However, an optimization of the method's time was accomplished.

Another big limitation is having to estimate manually the number of clusters. It would be much easier to use a method as mean shift that estimates the number of clusters automatically. This would save a lot of time with similar results.

Other limitation is that clustering is useful for segmentation (that is why our methods performed better in Jaccard index than in AP). Clustering might not be the best option for contour detection. Additionally, clustering creates closed sections, while other contour detection algorithms can produce open regions.

Another limitation is that our method produces hard clusters, with only one level of intensity. It would be more useful to have a family of borders with different probabilities of being border (as Pablo's method). Probably a better usage of gmm can be useful for obtaining decent results with clustering.

Other limitation is the creation of false contours, and the lack of an option for eliminating those contours. One easy solution could be to use normalized cuts, for removing similar boundaries, and thus, improving the method's precision. Another possibility is to use UCM (as Pablo) and find the optimal threshold for eliminating the majority of false contours without losing true positives.

A substantial improvement could be obtained considering the best channel of each color space. For example, from RGB if R is more discriminative than G and B, one should only keep R as a descriptor. Additionally, if the L channel of Lab showed a higher discrimination compared to a, b, G and B, it should be considered as another descriptor.

The idea is to build a multi-channel feature space with high discrimination, producing a better precision and recall performance.

Another improvement could be based on a pre-processing algorithm. This could be useful for high-computational cost methods (as gmm+XY) that produce high Precision-Recall metrics. The idea is to fit the images to look like a gaussian distribution, in order to fasten the process of making the segmentations. One possibility for this is applying a Gaussian filter in windows of nxn, and then run the method with the concatenation of each window. This should diminish the time of gmm+xy without affecting significantly the AP obtained (hopefully).

General final comments

RGB+XY remained as the method with best performance. However, it's time of execution is higher than HSV. Anyway, the relation between performance and time makes RGB+XY the best method. At the beginning of the paper HSV was the best method because of its fast implementation. Anyway, it's time advantage was overwhelmed by the RGB+XY performance.

RGB+XY might been way better than HSV because there were larger amount of images. In the previous work [1] there were way less images. One might think that RGB+XY is more robust compared with HSV.

It would be interesting to try optimizing gmm, but its computational cost is way to high. That was the reason why this method wasn't studied.

4. Conclusions

Clustering is a good segmentation algorithm, but it isn't too good as a contour detection algorithm. The fact of having closed areas gives it a better performance in segmentation than in contour detection.

One of the most expensive things is to find the correct k for obtaining better results. Most of the times with a very low k (5) one can have better results than with bigger k's. However, there are some exceptions in which a higher k can show good performance.

The image representation trough different feature spaces is vital for obtaining good results. A rich representation of the image (as textons with probability of boundary) will produce better results. Even, taking into account spatial information can give better results than only considering intensity or color. However, it is important to say that this method has the disadvantage of partitioning homogeneous regions just because they are far away.

Post segmentation/contour detection processes are also vital for obtaining good results. The use of normalized cuts (or its eigenvectors) can be really useful for reducing the number of false positives.

References

[1] Mejia, J. *Lab 06 - Segmentation*. Universidad de Los Andes. 2018.

[2] Arbelaez, P., Maire, M., Fowlkes, C., Malik, J. *Contour Detection and Hierarchical Image Segmentation*. Berkeley. 2010.