Computer Vision - Practicum 7

Authors: Emer Rodriguez Formisano and Jorge Alexander

Date: 27/11/2017

Introduction

Before starting to answer the questions of the practium definition, a brief explanation of the code implemented is done in order to clarify how the analysis was performed. As several steps must be repeated with different parameters and images, various functions have been defined to automate the process. There are 2 main functions used in the analysis called `runmethod` and `runparallelanalysis.`

The `runmethod` function has the code to run either k-means or MeanShiftCluster methods with a given number of paritions `k` or a bandwidth parameter `b` respectively. It has also two extra parameters, one enables the usage of the spatial coordinates for the analysis and the second enables the calculus of the mean color for each clustered regions. If both are `false`, the anlysis is done using only RGB colours and the output is presented with black and white regions.

The second function is `runparallelanalysis` which invokes `runmethod function` several times with different parameters in order to generate the same figure (Fig.1) in the practicum definition. As stated, the image contains:

- Original RGB image
- K-means cluster analysis using k partitions without spatial coordinates and regions in gray scale
- K-means cluster analysis using k partitions without spatial coordinates and regions in mean colour
- K-means cluster analysis using k partitions with spatial coordinates and regions in gray scale
- K-means cluster analysis using k partitions with spatial coordinates and regions in mean colour
- MeanShift cluster analysis using bandwidth parameter without spatial coordinates and regions in gray scale
- MeanShift cluster analysis using bandwidth parameter without spatial coordinates and regions in mean colour
- MeanShift cluster analysis using bandwidth parameter with spatial coordinates and regions in gray scale
- MeanShift cluster analysis using bandwidth parameter with spatial coordinates and regions in mean colour

There are 8 images to be generated, and the function generates them in parallel using the Parallel Computing Toolbox. If the toolbox is not available, the analysis can be performed sequentially by using the `runanalysis` function instead. The main bottleneck of the execution is the MeanShift function as the k-means runs quite fast. The MeanShift initialize the windows with image pixels several times and runs the algorithm until convergence. This is computationally more expensive than running k-means.

The code of the function is presented at end of the report. The following code initialises the pool of workers.

```
pool = parpool('local', 4);
```

```
Starting parallel pool (parpool) using the 'local' profile ...
```

```
connected to 4 workers.
```

The report will contain the analysis of 3 pictures: Animals, Alvin and Big Bang Family. The analysis of the Animals will be more exhaustive as it will also answer the generic questions about the comparison of the two methods. The Analysis of the two other images will be focused on the differences due to the images itself using the best parameters found.

Animals picture

Image is loaded as follows.

```
animals_rgb = imread('images\animals.jpg');
```

What are the clustering algorithms parameters and what they mean

The main parameters of the k-means method are the data and the number of partitions k. Similarly, the main parameters of the MeanShift are the data and the bandwidth. The k-means tries to find k groups of pixels by grouping them with similar colour and spatial position if available. It could be interpreted as the number of regions or segments to be identified in the picture.The higher the value the more partitions will be found.

The MeanShift algorithm tries to group the pixels as well but taking into account the disitrbution of the points in the space. Instead a predefined number of groups, it uses a bandwidth paramter which is the size of the window in order to search for modes (regions with high density). The higher the bandwidth, the less number of clusters will be detected and the smoother the segments of the image will be.

Find optimal parameters (k: number of partitions and b: bandwidth coefficient)

For this task, 3 scenarios were considered:

- Scenario 1 with k = 2 and b = 120
- Scenario 2 with k = 4 and b = 60
- Scenario 3 with k = 8 and b = 40

Scenario 1 with k = 2 and b = 120

This is an extreme scenario with too few segments. The k means clustering splits the pixels in a binary group, bright and dark pixels. That is clear when the the coordinates are not considered. If the spatial information is added, the group is split from the middle of the image as the pixels on the right hand side are brighter.

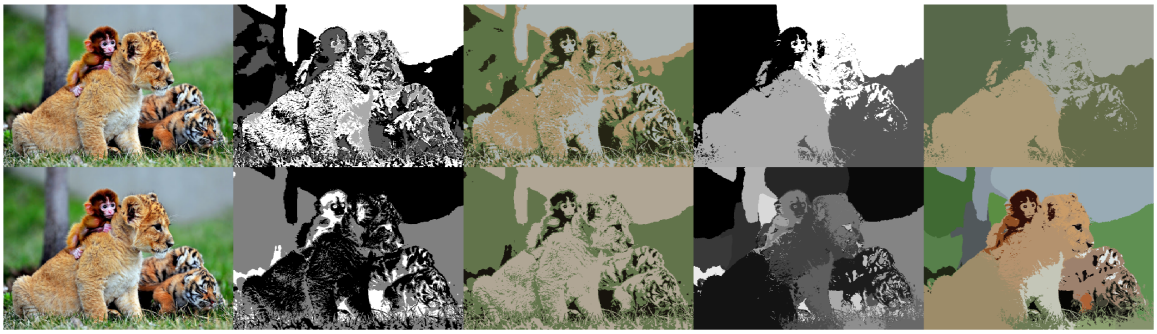
The MeanShift also performs very poorly too because the window is too wide. The runs without the spatial information only detect 1 cluster, which is useless. However, if the coordinates are considered, it detects 4 main areas. One area contains the almost the 4 animals together and the rest are background pixels grouped by intensity.

```
imshow(runparallelanalysis(animals_rgb, 2, 120))
```

Scenario 2 with k = 4 and b = 60

Reducing the window to the half and doubling the number of partitions provides more sensible results. The clustering without the coordinates shows similar results for both methods as they group the pixels by intensity using 4 and 3 groups. However, if the spatial information is added, MeanShift generats much more regions. There are more than a dozen but the shape of the animals can be distinguished. However, each animal is composed by several groups as well as the background. K-means tries to do its best with just 4 groups and the background and animals are clearly mixed.

```
imshow(runparallelanalysis(animals_rgb, 4, 60))
```



Scenario 3 with k = 8 and b = 40

The last scenario considers the parameters that focus too much into the image details. The number of paritions is 8 and the window size is 40. Without using coordinates, the K-means tried to do an 8 bit convertion of the original image. MeanShift has done it better but the 4 animals still forms a single group. Adding the spatial coordinates help the MeanShift to separate the animals but now they are composed by too many groups. The lion is made of at least 8 different groups of pixels. The background was also divided into several parts.

```
imshow(runparallelanalysis(animals_rgb, 8, 40))
```

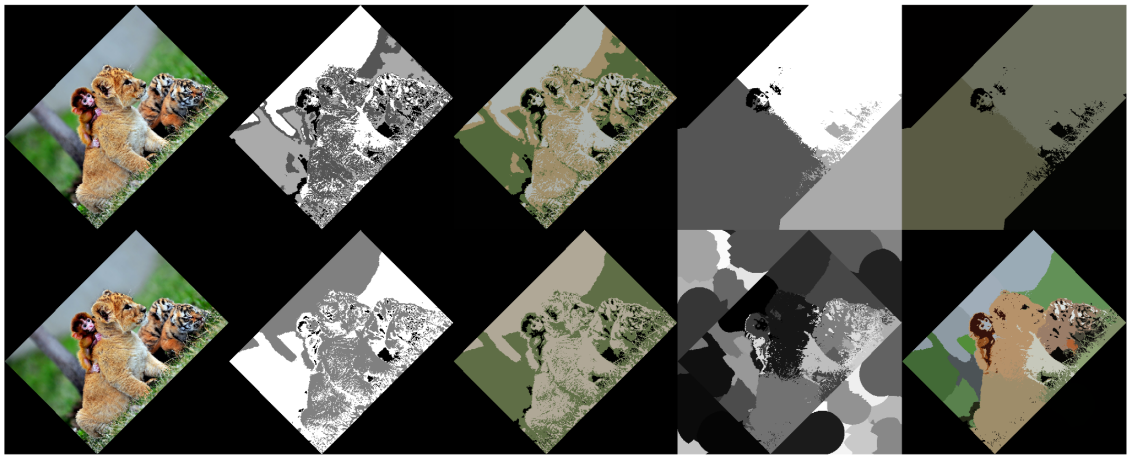
Results when adding image modifications.

Considering the settings of scenario 2, a series of tests were done in order to measure the impact of image rotation and rescale.

Rotation

```
animals_rgb_45deg = imrotate(animals_rgb, 45);
imshow(runparallelanalysis(animals_rgb_45deg, 4, 60))
```

```
Warning: Image is too big to fit on screen; displaying at 67%
```

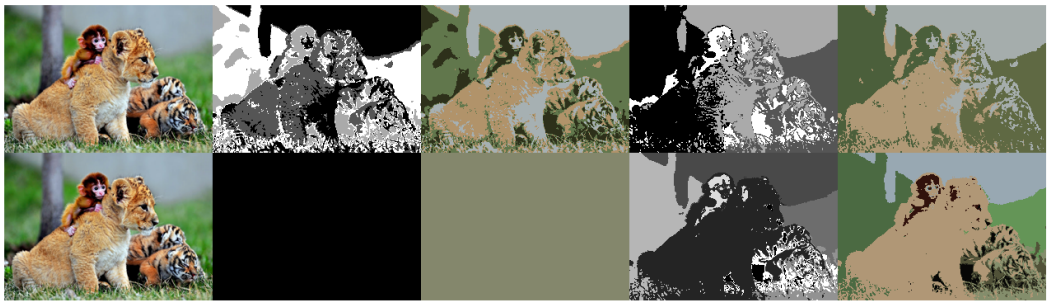


As shown in the previous image, the rotation impacts negativelly the analysis, specially when the spatial information is used. The "empty" margins are pixels with value of 0 which form a colour group when the coordinates are not considered. If the coordinates are instead considered, the margin pixels information is combined with pixels of the image. As MeanShift method uses the data distribution, its results are more robust.

Scaling the image down

```
animals_rgb_scaleddown50 = imresize(animals_rgb, 0.5);
```

```
imshow(runparallelanalysis(animals_rgb_scaleddown50, 4, 60))
```



The results after scaling the original image down have lower quality. In other words, the results obtained using the settings of the scenario 2 with a scaled down picture are similar to the ones obtained with the scenario 1 which used poor settings. Note how MeanShift method was unable to make groups when the coordinates were not considered. However, the k-means results are more robust to image resizing.

Reproducibility of results

Both methods use randomness in order to perform the analysis and as they converge to local optimums, the results will be different every run.

In the case of the k-means, the initial k centroids are randomly located. The following picture shows 3 different executions of the alogrithm with the same settings and the results are quite consistent. The differences are usually detected at cluster's frontiers.

```
rr1 = [runmethod(animals_rgb, 4, 0, true, true) runmethod(animals_rgb, 4, 0, true, true) runme
imshow(rr1)
```

In the case of MeanShift, a random pixel is used as a starting mean. MeanShift is more sensitive to the randomness as the 3 repeated executions with the same settings show more differences than the k-means example. For instance, the right ear of the lion is less defined in the second run. There is a brown patch on the bottom right of the pictures from the first and second run, which was not obtained in the third run.

```
rr2 = [runmethod(animals_rgb, 0, 60, true, true) runmethod(animals_rgb, 0, 60, true, true) ru
```

```
imshow(rr2)
```

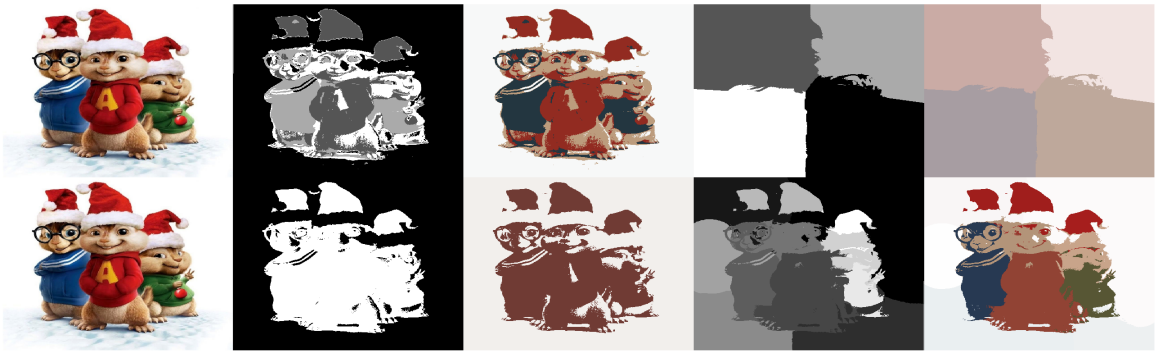
Although there are some differences, the overall results are quite consistent. The reproducibiliy of the examples can also be ensured by setting a fixed seed before the executions.

Alvin 2 picture

After some trial and error the following settings provides a fair result for the Alvin 2 picture analysis.

```
alwin2_rgb = imread('images\alwin2.jpg');
imshow(runparallelanalysis(alwin2_rgb, 4, 150));
```

```
Warning: Image is too big to fit on screen; displaying at 50%
```



Apparently, the Alvin 2 picture should be easier to segment. The elements are differentiable as the t-shirts are in R, G, B. The MeanShift method does indeed intentify the bodies of each chipmunks. However, the faces are a bit mixed. The white ribbon of the Chrismas hat is also mixed with the background. The K-means result with coordinates is not comparable with the one obtained with MeanShift.

It was also detected that the parameter of bandwidth of the MeanShift method is dependent on the image size. The bigger the image, the bigger the bandwidth parameter (window) should be used. If the window size or bandwidth is defined relatively to the image size, this issue could be avoided.

Big Bang family picture

The report will conclude with the analysis of the Big Bang family picture. After some trials, the following settings returns a sensible result.

```
bigbang_rgb = imread('images\bigbangfamily.png');
imshow(runparallelanalysis(bigbang_rgb, 4, 60));
```

Again, MeanShift outperforms kmeans method. The last MeanShift execution with coordinates indentifies the different characters. However, some difficulties are found when some characters overlap others. For instance, Amy's skirt is mixed with Sheldon's and Leonard's trousers due to the colour similarities. The coloured clothes clearly helps to separate each one.

Defined functions

```
function [imres] = runmethod(im, k, b, usecoord, usemeancolour)
    % Function can perform the analysis of any combination of following
    % propertites:
    %  - Using k-means or MeanShiftCluster method
    %  - Using spatial coordinates or not
    %  - Using the mean colour of the cluster or not

    % Prepare Matrix of features
    im1d = reshape(im, size(im, 1) * size(im, 2), 3);

    if(usecoord)
        % Add spatial coordinates
        coord_y = repmat(1:size(im, 2), size(im, 1), 1);
        coord_y = coord_y(:);
        coord_x = repmat(1:size(im, 1), 1,size(im, 2))';

        im1d = [double(im1d), coord_x, coord_y];
    end

    % Method
    if(k > 0)
        idx = kmeans(double(im1d), k);
    elseif(b > 0)
        [~, ms_idx, ~] = MeanShiftCluster(double(im1d'), b, false);
        idx = ms_idx';
```

```
    end

    % Reshape cluster index into image dimensions
    im2d = reshape(idx, size(im, 1), size(im, 2));

    % Apply colour
    if(usemeancolour)
        % Mean colour
        im2dres = zeros(size(im, 1), size(im, 2), 3);
        for cl = unique(idx)'
            idx_class = find(idx==cl);
            for ch = 1:3 % rgb
                idx_ch = idx_class + (ch-1) * (size(im,1) * size(im,2));
                im2dres(idx_ch) = mean(im(idx_ch));
            end
        end
        im2dres = uint8(im2dres);
    else
        % Black and white
        im2d = ((double(im2d) - min(min(im2d)))/ ...
            ((max(max(im2d)) - min(min(im2d)))+1e-16)* 255.0);
        im2dres = cat(3, im2d, im2d, im2d);
    end

    imres = im2dres;

end
```

```
function [imres] = runanalysis(im, k, b)
    % k-means row
    km_nocoord_gray = runmethod(im, k, 0, false, false);
    km_nocoord_colo = runmethod(im, k, 0, false, true);
    km_xycoord_gray = runmethod(im, k, 0, true, false);
    km_xycoord_colo = runmethod(im, k, 0, true, true);

    km_row = horzcat(im, km_nocoord_gray, km_nocoord_colo, ...
        km_xycoord_gray, km_xycoord_colo);

    % MeanShift row
    ms_nocoord_gray = runmethod(im, 0, b, false, false);
    ms_nocoord_colo = runmethod(im, 0, b, false, true);
    ms_xycoord_gray = runmethod(im, 0, b, true, false);
    ms_xycoord_colo = runmethod(im, 0, b, true, true);

    ms_row = horzcat(im, ms_nocoord_gray, ms_nocoord_colo, ...
        ms_xycoord_gray, ms_xycoord_colo);

    imres = [km_row; ms_row];

end
```

```
function [imres] = runparallelanalysis(im, k, b)

    p = gcp('nocreate');

    % Generate matrix of parameters
    design_mat = combvec([0,1],[0,1],[0,1])';
    design_mat = [~design_mat(:,3)*k,design_mat(:,3)*b,design_mat(:,[2,1])];

    % Invoke runmethod executions
    for i = 1:size(design_mat,1)
        f(i) = parfeval(p, @runmethod, 1, im, design_mat(i,1), ...
            design_mat(i,2), design_mat(i,3), design_mat(i,4));
    end

    % Collect results as they become available
    results = cell(1,length(f));
    for i = 1:length(f)
      [completedIdx,value] = fetchNext(f);
      results{completedIdx} = value;
    end

    % Compose final image
    imres = [im, results{1:4}; im, results{5:8}];

end
```