2017

Video segmentation & Texture extraction

COMPUTER VISION – LAB 9/10

JOSEP FAMADAS ALSAMORA / JORDI RIU VICENTE

UNIVERSITAT DE BARCELONA

UB | Facultat de Matemàtiques i Informàtica

Table of contents

Table of Figures

1. Video Segmentation

1.1. Background Subtraction

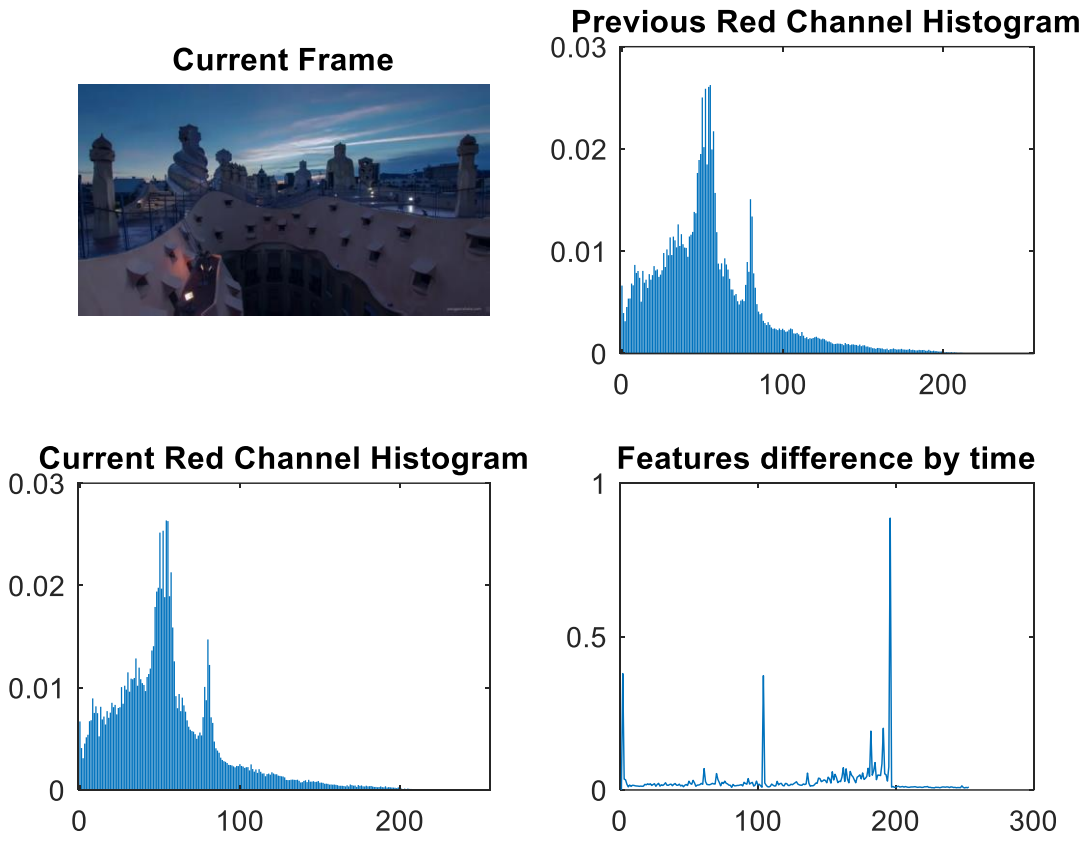

Figure 1: Features difference evolution by time.

To compute the features difference between 2 consequent frames, first we compute the normalized histogram for each one of the color channels of the frame. Once this is done, the features difference between frames is computed as:

$$diff(frame\ i, frame\ i+1) = \sum_{j \in \{Red,Green,Blue\}} \left\| hist_j(frame\ i) - hist_j(frame\ i+1) \right\|$$

That is, summing for all color channels the norm of histograms differences for each channel.

Once the Feature difference vector is computed, and by inspection, we establish a threshold value (th) for which if $diff(frame\ i, frame\ i+1) > th$ then both frames are considered to belong to different shots. Specifically, we set th = 0.25.

In Figure 2, we observe that, for some parts of the video, the peaks in the features difference plot that correspond to shot changes are not clear. Specifically, this happens when the shot under study has a fast-changing background (for example the shot in which the camera records

from inside a fast-moving car). This is one of the limitations that we expected to find for this algorithm.

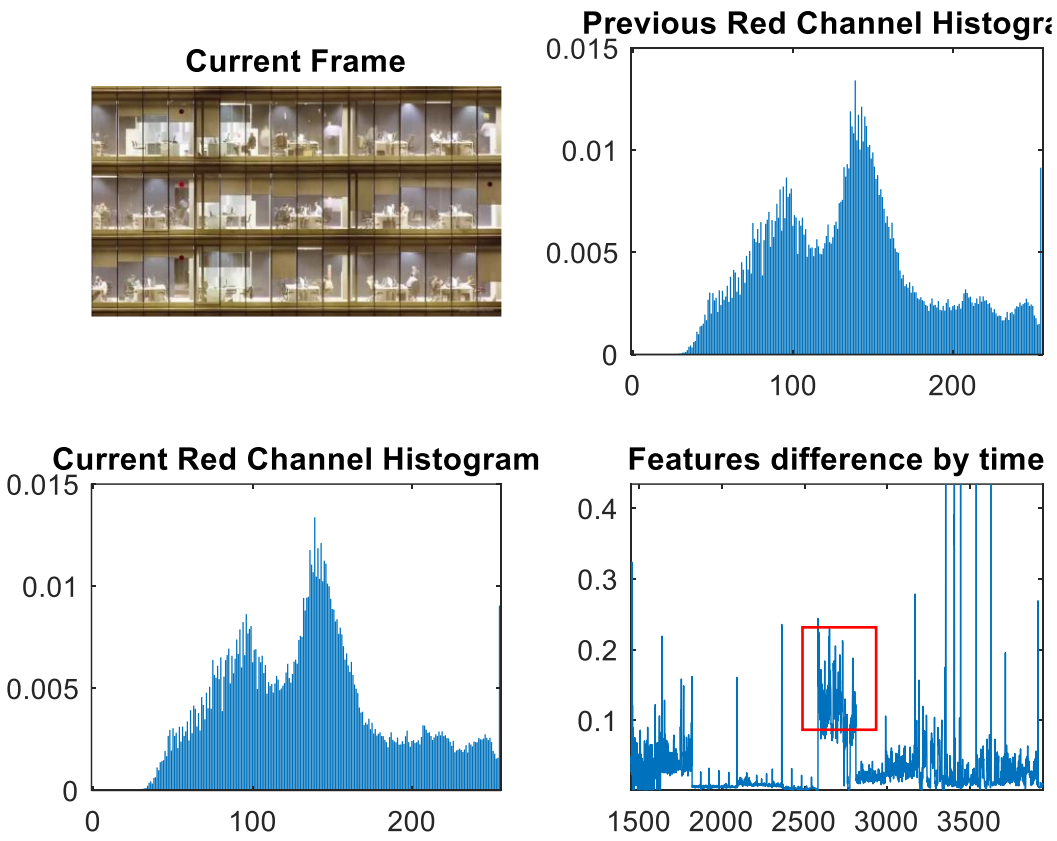

Figure 2 Shot with a fast-moving background.

Once each shot has been selected, we apply the median filter to substract the background of the image for each shot. In Figures 3, 4 & 5 we observe the results. For the first one, only the sun moves during the shot, and therefore, it is detected as a segmented object.

Figure 3: Median Filter applied to the first shot. Left: Background Extracted, Center: Current Frame, Right: Segmented Objects

For the second one, the only objects which position is evolving during the shot are the clouds:

Figure 4: Median Filter applied to the second shot under study. Left: Background Extracted, Center: Current Frame, Right: Segmented Objects

Finally, for the last shot, we observe that the background of the shot consists on the city of Barcelona before the lights are turned on. Therefore, all illuminated regions are obtained as segmented objects.

Figure 5: Median Filter applied to the third shot under study. Left: Background Extracted, Center: Current Frame, Right: Segmented Objects

We have also tried applying another background filter which consisted on defining the value for each pixel in the background shot as the mean value of that pixel for all frames within the shot. The obtained results were slightly worse the ones obtained with the median filter (See Figure 6).

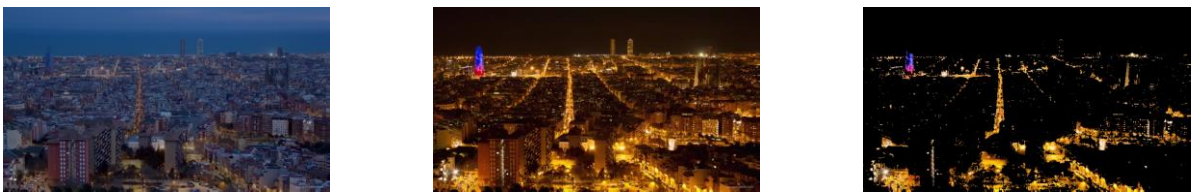

Figure 6: Mean Filter applied to the third shot under study. Left: Background Extracted, Center: Current Frame, Right: Segmented Objects

Finally, we applied the last background filter which consisted on defining the value for each pixel in the background shot as the most frequent value of that pixel for all frames within the shot. The obtained results are considerably worse (See Figure 7).

Figure 7: Most Frequent Value Filter applied to the first shot under study. Left: Background Extracted, Center: Current Frame, Right: Segmented Objects.

1.2. Video Segmentation for event extraction

To divide the provided images between indoor and outdoor events we took the following approach:

The most relevant features differences we saw between indoor and outdoor images was the amount of "sky" pixels within an indoor image and an outdoor one.

To categorize a sky pixel, we used the following value ranges for each color channel of the image:

$$Red \in [110,190]\ ,\quad Green \in [110,190],\ Blue \in [220,255]$$

Moreover, we also forced that, for a sky pixel:

$$|Red\ Value - Green\ Value| < 40$$

We obtained such rules by manually inspecting the RGB values of sky pixels.

Considering these rules, we plotted the number of pixels for indoor images and outdoor images (See Figure 8):

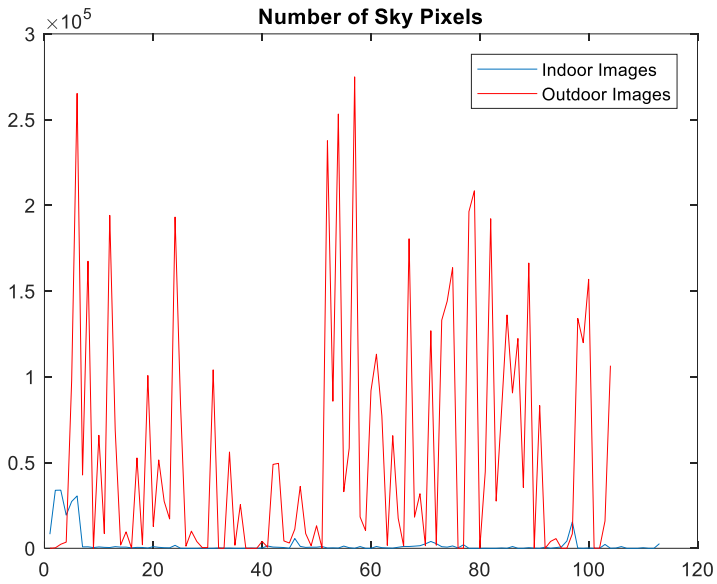

Figure 8: Number of sky pixels for indoor (blue) and outdoor (red) images.

We clearly observe that for indoor images the number of sky pixels is much lower.

Finally, we define a threshold such that if the number of sky pixels in an image is higher than such threshold, then the image is categorized as outdoor. Specifically, the value for the threshold is 1.37x10^3. Such value is the one that maximizes the accuracy of the algorithm for these set of images.

The obtained accuracy of the classification under these circumstances is **81.57%.**

2. Texture descriptors extraction

In this second part of the deliverable we have focused on classifying images using their textures.

2.1. Descriptors generation

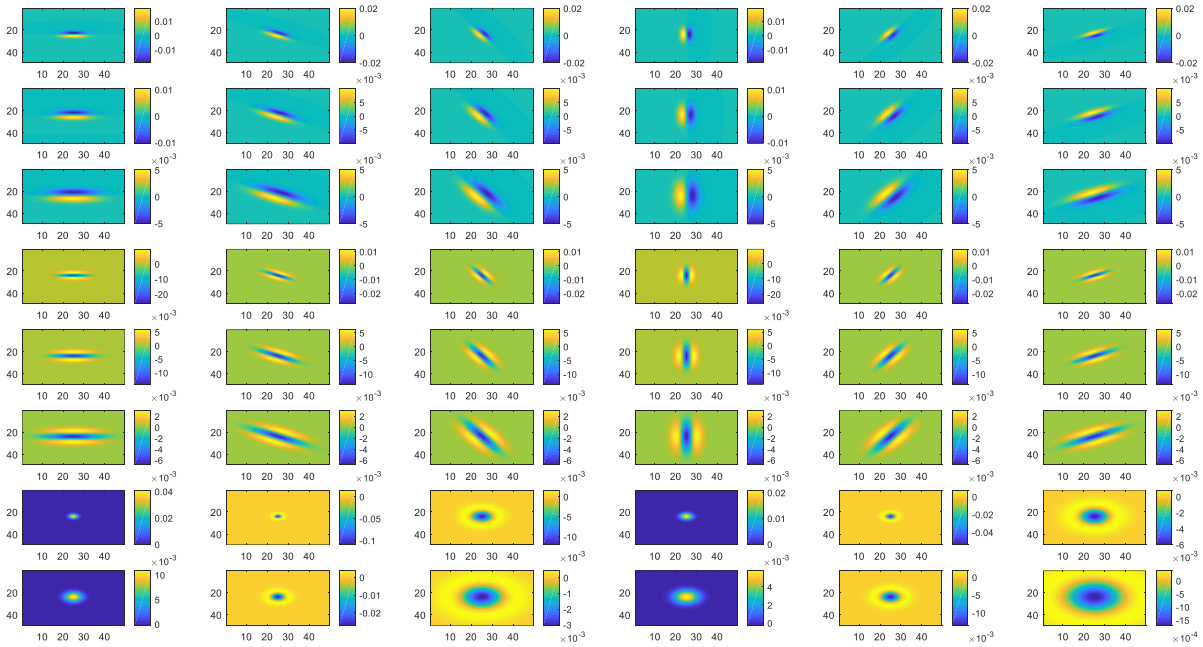

Figure 9: LM Filters

In *Figure 9* we can see the 48 LM filters. They consist of:

- 18 edge detectors in angles from 0º to 150º in steps of 30º and in 3 different sizes.
- 18 laplacian filters to detect bars with the same angles of the edge detectors.
- 12 gaussian filters to detect spots of different size.

When we want to extract the descriptors of an image, the 48 filters are convolved with the image (previously transformed to grayscale) and the mean value of the result is calculated, giving as a result a horizontal vector with 48 elements corresponding to each of the filters.

This can be extended to a group of images giving as a result a matrix of size [num_im x 48].

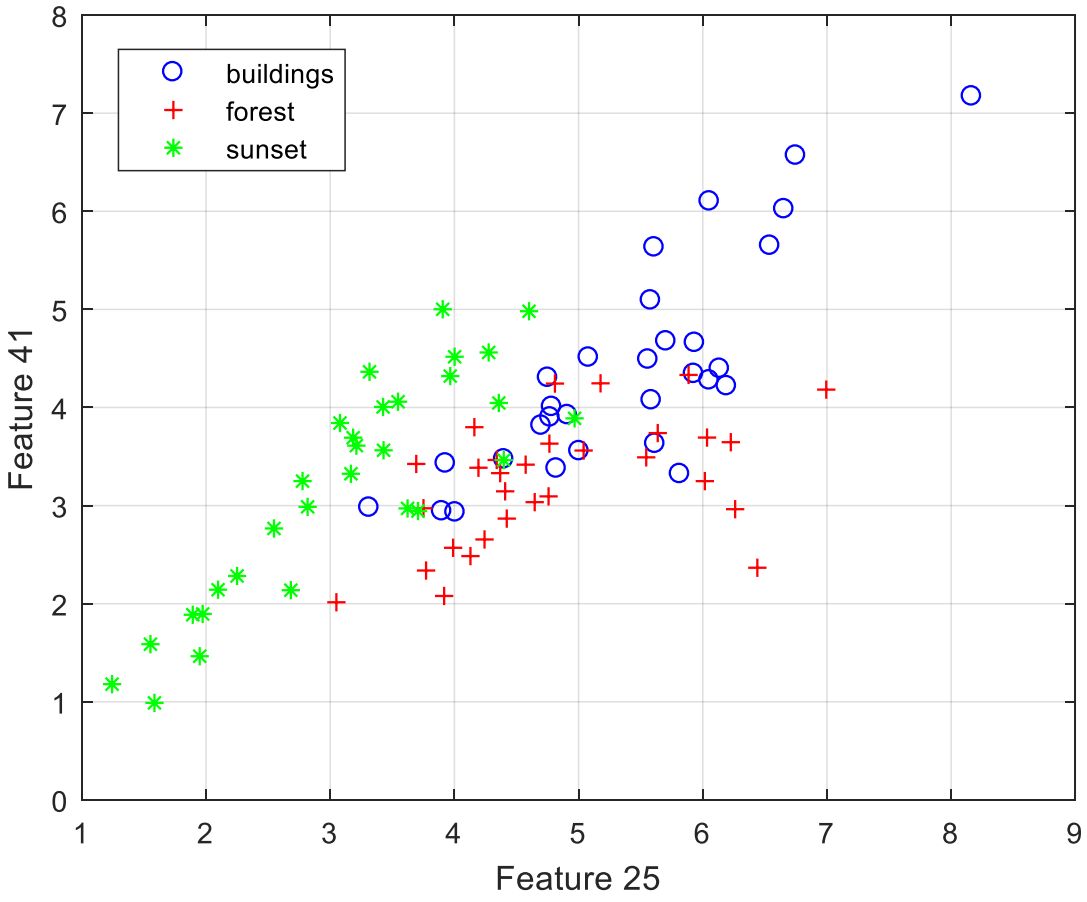

Figure 10: Features 41 and 25 from all the images.

In *Figure 10* it is plotted the feature 41 with the 25 of 90 different images: 30 corresponding to buildings (blue), 30 to forest (red) and 30 to sunsets (green). As it can be observed, the buildings and the sunset images can be easily differentiated using this feature. However, we will be using the 48 features.

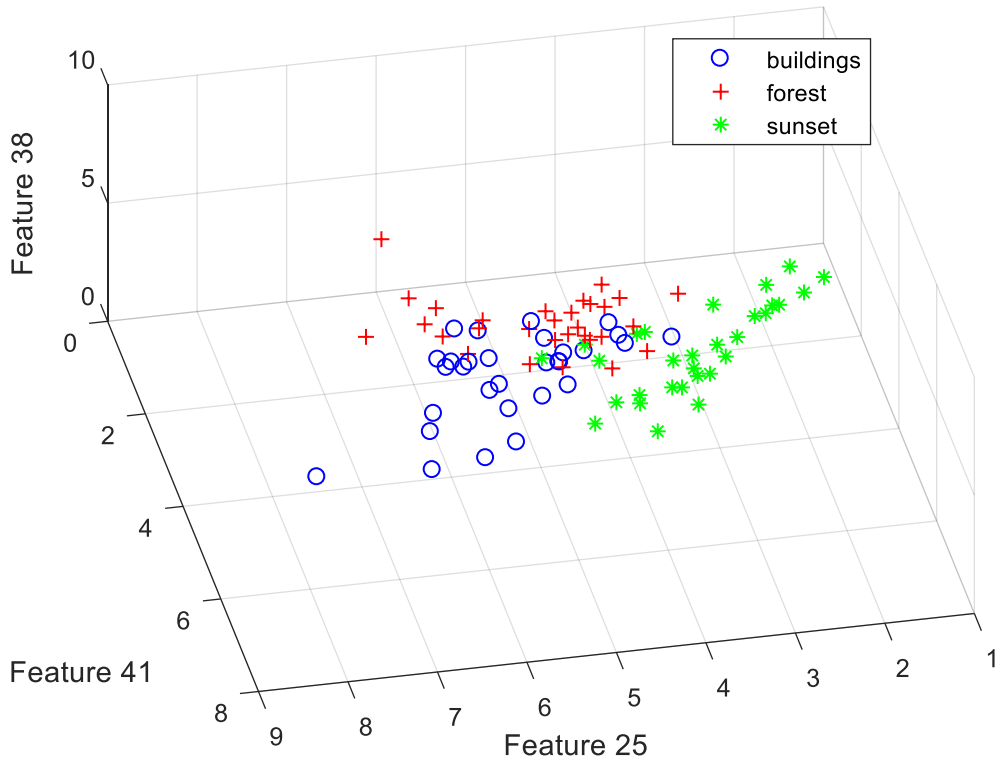

Figure 11: Features 41, 25 and 38 from all the images.

In *Figure 11* the plot is done in 3D adding the 38th feature. Here the 3 different image types can easily be differentiated.

2.2. Finding similar image

Now, having the 90x48 matrix the objective of this part was to, given an image as a query, find the 'k' most similar images from the other 89. In our work we have decided to use k = 9.

Figure 12: 9 More similar images to the Image Query (Building_01).

In *Figure 12* it can be appreciated that, when given as an input an image of a building, the system detects as the 9 most similar images 8 buildings and 1 forest. Seeing these results, we can conclude that this is a good classification system for buildings, which have really good defined textures.

However, in *Figure 13* we can see that for images of sunsets, where the color is far more important than the shapes, the system does not work so well. In this particular case, it will assign the image to the category of forest.

Figure 13: 9 More similar images to the Image Query (Sunset_13).

In order to avoid this, the color information can be taken into account. To do this, instead of transforming the image to grayscale and calculate the 1x48 descriptor vector, the vector is computed three times, one for each of the RGB channels and then concatenated horizontally giving a 1x144 descriptor vector. The process with all the images of the directories is performed the same as before.

Taking now into account the 3 RGB descriptors in *Figure 14* can be appreciated that the result is far better.

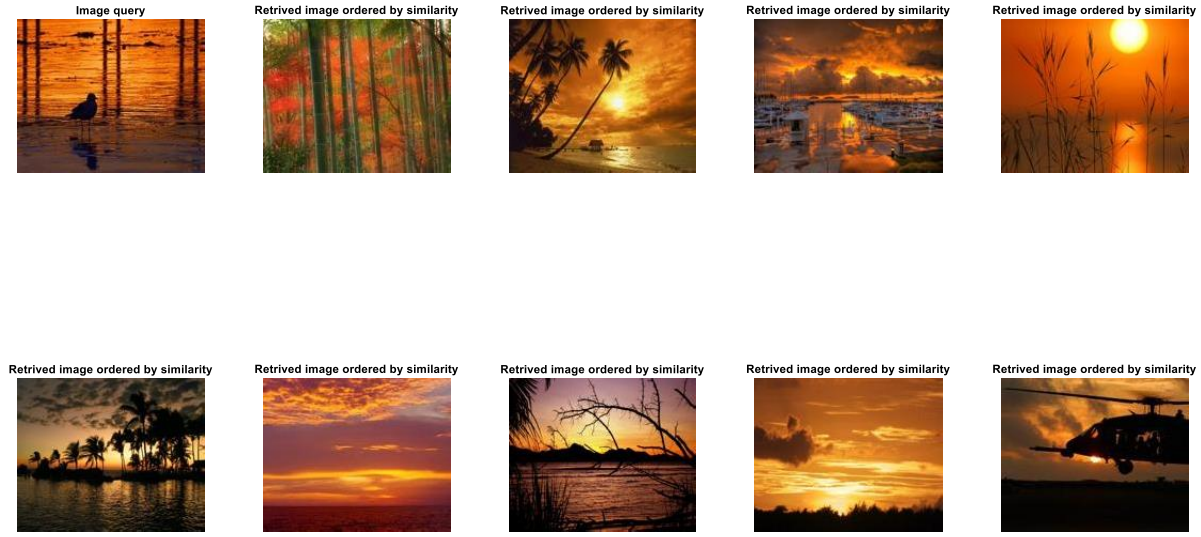

Figure 14: 9 More similar images to the Image Query (Sunset_13) using color information.