Personal project: Recognising and classifying the position of some solar protections via image recognition with machine learning

Marc Roca-Musach

8th January, 2023

Abstract

In the context of the HarvardX Data Science MOOC we learned some machine learning concepts and tools. My final project wants to put in practice what I learned. I developed the following project with a personal data set of images of a building's façade that has some operable solar protections. The aim of this project is to identify the position of the solar protections of every window using a machine learning algorithm.

Contents

1 Introduction

Machine learning is specially useful when we have to analyze thousands of data. It can also be useful to identify properties from a graphical source (computer vision). In this final project I practice how to use machine learning on a graphical set of images where we need to identify some characteristics on it.

This project is personally challenging as I assume it as an step forward from what we were taught in the course series. The most similar exercise we practiced in the course was the one to distinguish 2 or 7 from a hand-written set of numbers. In that exercise, we used black&white pixel information as features to train some basic machine learning algorithm. This projects starts form this concept, but it faces many new challenges such as image processing, the distortion produced by color, clustering algorithms, and untagged learning. To solve them, I self-learned some new libraries not provided in the course.

1.1 Data source

The data set used in this project comes from a personal source. In order to make it available for the project review, it is uploaded on Github. It consists of a sequence of images of the West façade of a building. This façade comprises a regular matrix of windows, each one with an outdoor solar protection system. The photos were taken every hourly for a period of two weeks, always from the same angle and position of the camera.

In total there are 336 images, each one containing 163 windows. In total, there are 54768 windows.

Figure 1: Sample image of the data set.

1.2 Aim

The aim of this project is to identify the position of the solar protections of every window of the façade using a machine learning algorithm.

2 Methods and analysis

Steps 1 and 2 correspond to the preparation of the images. Step 3 is the definition of the features required for machine learning algorithm. Steps 4 and 5 are two different methods for clustering based on the features. Step 6 corresponds to the tagging process.

2.1 Image processing

The images from the dataset were taken 24/7. This means that the lighting conditions change regularly (from night to day, from backlight to indirect lighting), as shown in the following images:

This first step of the method tries to enhance contrast and the tone of different materials to simplify the detection of differences with the machine learning algorith. This process is executed for all the images by applying a normalization method and a modulation method form the *magick* package. The previous images edited are the following:

2.2 Crop the position of every window

We want to obtain the position of the solar protection for every window. This means that, first, we need to crop every window from every image and save it as a new image. This process was automated by defining the position of the top-left window (variable `img_window1_crop`), the position of the bottom-right window (variable `img_windowN_crop`), the amount of windows in the X and Y direction (variables `window_quantity_x` and `window_quantity_y`) and the size of the windows in the image (variables `window_width` and `window_height`).

This project accounts for a defaults section in the *Rscript* file that simplifies the execution. However, this parameters were initially adjusted manually to fit the input images.

```
# DEFAULTS
if(!exists("facade")) { facade <- "West" }

if(!exists("img_max_width")) { img_max_width <- 1000 }

if(!exists("img_window1_crop")) { img_window1_crop <- "10x33+15+54" }
if(!exists("img_windowN_crop")) { img_windowN_crop <- "12x34+375+212" }

if(!exists("window_quantity_x")) { window_quantity_x <- 28 }
if(!exists("window_quantity_y")) { window_quantity_y <- 6 }

if(!exists("window_width")) { window_width <- 12 }
if(!exists("window_height")) { window_height <- 33 }
```

I generate a table of windows, indicating all the data related to the windows' position, name and crop parameters.

Table 1: First 6 rows of the `windows` table

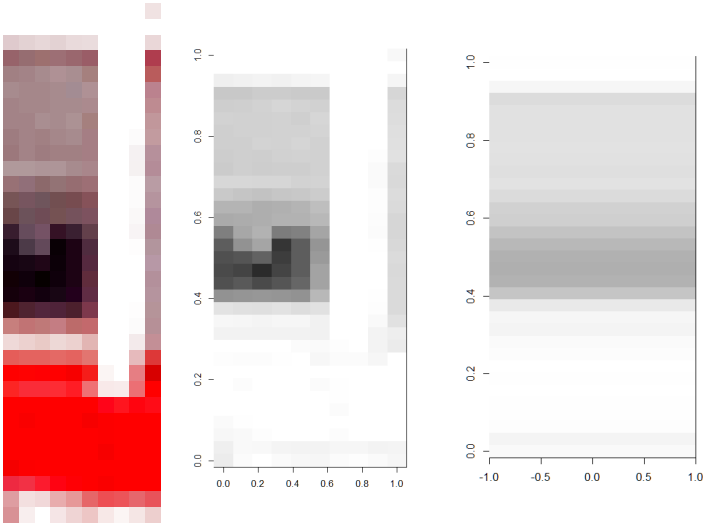
Name	center_x	center_y	crop
W601	21	70	12x33+15+54
W602	34	70	12x33+28+54
W603	48	70	12x33+42+54
W604	61	70	12x33+55+54
W605	74	70	12x33+68+54
W606	88	70	12x33+82+54

Then, the data in this table is used to massively crop all windows in all images and save them in a new folder.

2.3 Defining the features of the windows

The features of each window are defined as the darkness of every pixel in the red channel. I pick up the red channel because, after the modulation of the images at the first step, light gets represented in cyan, and darkness in red or black.

Every window is a picture of 12 by 33 pixels. This means a total of 396 pixels per window. The darkness of each pixel is read and stored in association with the ID of the window and the time stamp of the image. However, using every pixel as a feature could mean to overload the algorithm. I propose a simplification of the number of features: The desired value of the solar protections is the percentage of openness, equivalent to the percentage of daylight getting inside through the window. These solar protections open by sliding along some vertical rails. This mean the position of the solar protections is always horizontal. Thus, to simplify the amount of features, I sum up the values by row, moving from 396 features to 33 per window.



This is a sample of the first 15 features for the first 6 windows. Values range from 0 (black) to 255 (white).

Table 2: Sample of the features for the first 6 windows. Values range from 0 (black) to 255 (white).

ID	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15
20220930-0015_W106	229	225	230	239	232	228	228	225	221	232	255	162	83	106	138
20220930-0015_W107	231	228	234	235	232	232	229	231	231	254	255	255	106	85	85
20220930-0015_W108	234	232	222	223	226	231	234	241	240	246	252	184	162	166	183
20220930-0015_W109	241	241	240	243	242	239	235	236	237	251	236	236	237	233	232
20220930-0015_W110	227	211	197	209	215	217	219	218	203	228	224	224	223	221	220
20220930-0015_W111	220	214	211	213	212	210	207	200	181	209	212	215	213	207	201

And this is the distribution of darkness for the first image:

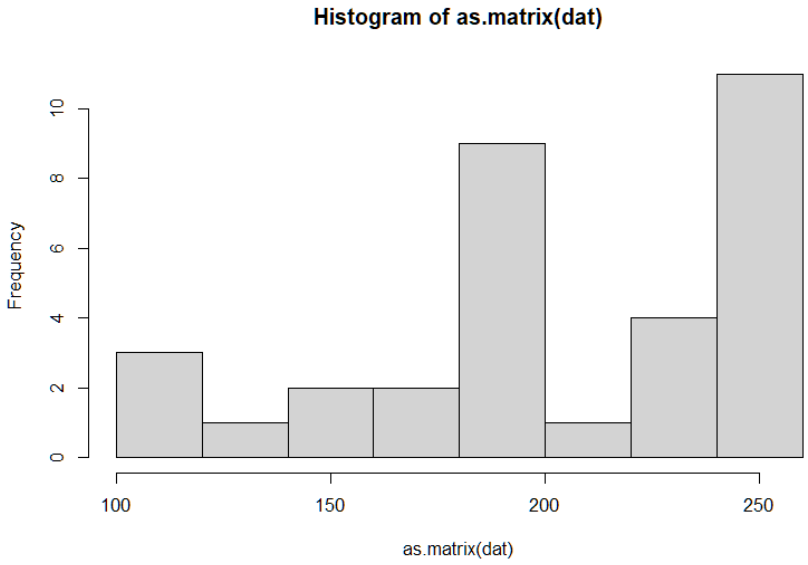


Figure 2: Histogram of darkness from the first window.

2.4 Clustering: Method 1 – The Kmeans algorithm

More information: https://en.wikipedia.org/wiki/K-means_clustering

Once defined the features at the previous step, the first clustering method used is the KMeans algorithm under the `stats` package. This algorithm clusters data according to a number of target clusters. For this reason, I initially have to decide the number of clusters. To do so, I use the `FitKMeans` function provided by the `useful` package. This function analyze sequentially the Hartigan's Number of a given number of clusters. If it is greater than 10, it considers that a new cluster should be added and recalculates the Hartigan's Number for the n+1 clusters. This process terminates when the `max.clusters` value is reached.

This is the code to calculate the number of required clusters. Seed number is a random number, but is fixed in order to ensure that always returns the same values.

```
# Get the data
features <- windows_features %>% select(-ID)

# Optimize the number of clusters based on the Hartigan's rule
clusters_fit <- FitKMeans( features ,
                           max.clusters=100 ,
                           nstart=25 ,
                           seed=45357452 ,
                           iter.max = 10 )

clusters_quantity <- clusters_fit$Clusters[ which( clusters_fit$AddCluster == FALSE )[1] ]
```

I obtain that the required number of clusters is 77.

Then, I run clustering process with the desired number of clusters:

```
km <- kmeans( features , centers = clusters_quantity , nstart = 25 , iter.max = 100000 )
```

2.5 Clustering: Method 2 – A model based approach

The second method is based on the model based approach provided by the `mclust` package. The `Mclust` function selects the optimal model via an expectation-maximization algorithm initialized by hierarchical clustering for parameterized Gaussian mixture models.

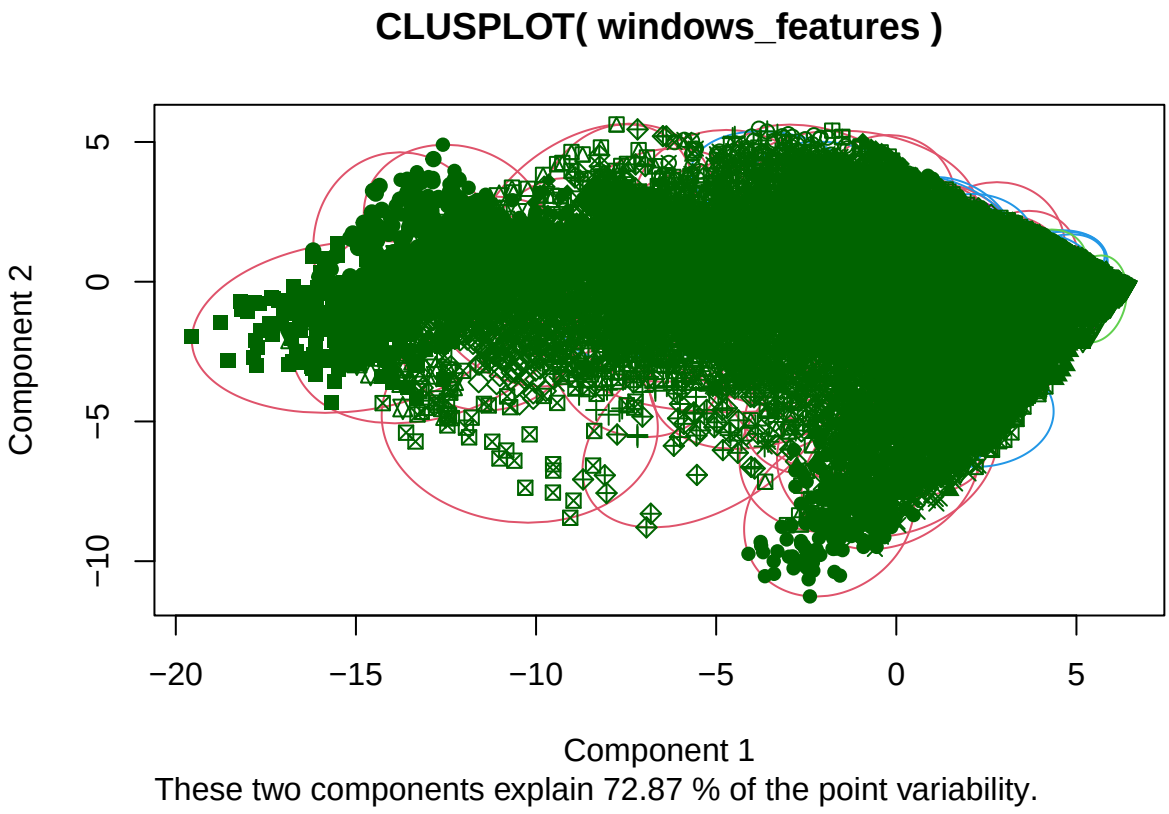
The model is fitted with the following code. I use the same amount of clusters as in the previous method to be able to compare how well do they work:

```
# Model Based clustering
mb <- Mclust( features , G = clusters_quantity )
```

3 Results

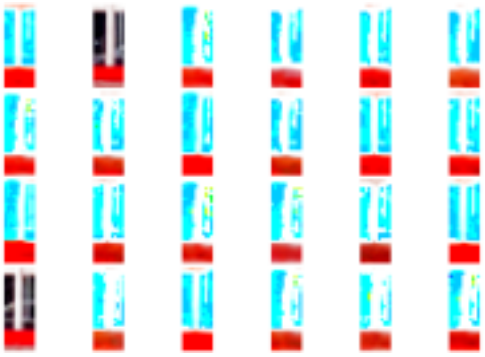
3.1 Results for method 1 – The Kmeans algorithm

The Kmeans algorighem clustered the images in 77 clusters. There are 33 features per window, which means that the criteria of each cluster is difficult to plot. However, the image below corresponds to the clusters by the 2 most relevant characteristics.

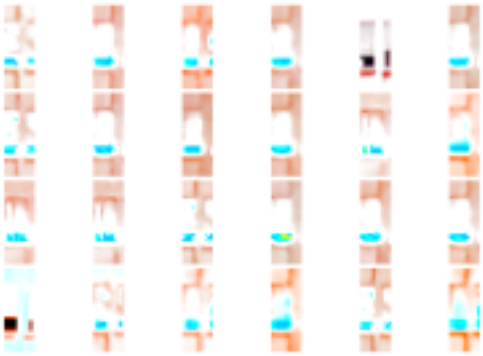


To evaluate whether the algorithm is clustering images according to the desired aim, the images below correspond to a random sample of 24 images from 4 of the clusters. We can see that all the images in each group have approximately the same position of the solar protections.

Sample of cluster number 2 (contains 93 elements in total):

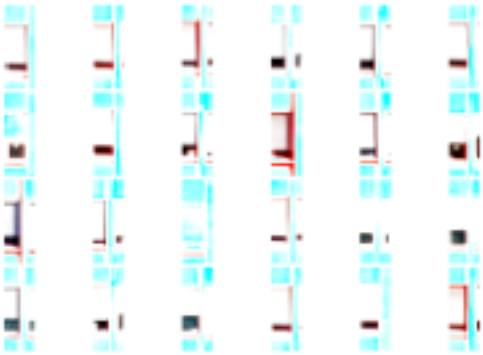


Sample of cluster number 14 (contains 165 elements in total):



Sample of cluster number 22 (contains 264 elements in total):

Sample of cluster number 63 (contains 274 elements in total):

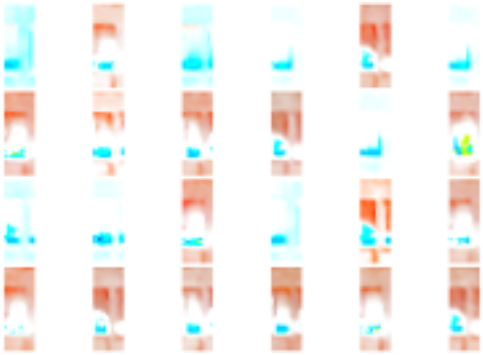


3.2 Results for method 2 – A model based approach

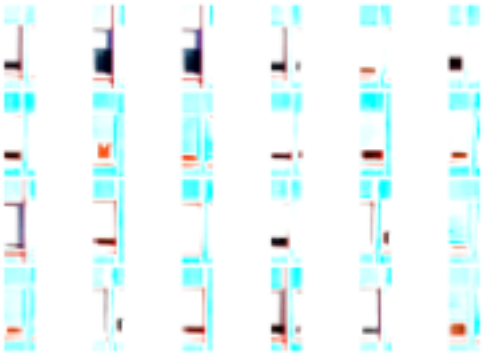
The model based approach was clustered with the same amount of clusters as in the Kmeans algorithm, to make them comparable (77 clusters). To evaluate whether the algorithm is clustering images according to the desired aim, the images below correspond to a random sample of 24 images from 4 of the clusters. We can see that all the images in each group have approximately the same position of the solar protections.

Sample of cluster number 2 (contains 163 elements in total):

Sample of cluster number 19 (contains 246 elements in total):



Sample of cluster number 22 (contains 430 elements in total):

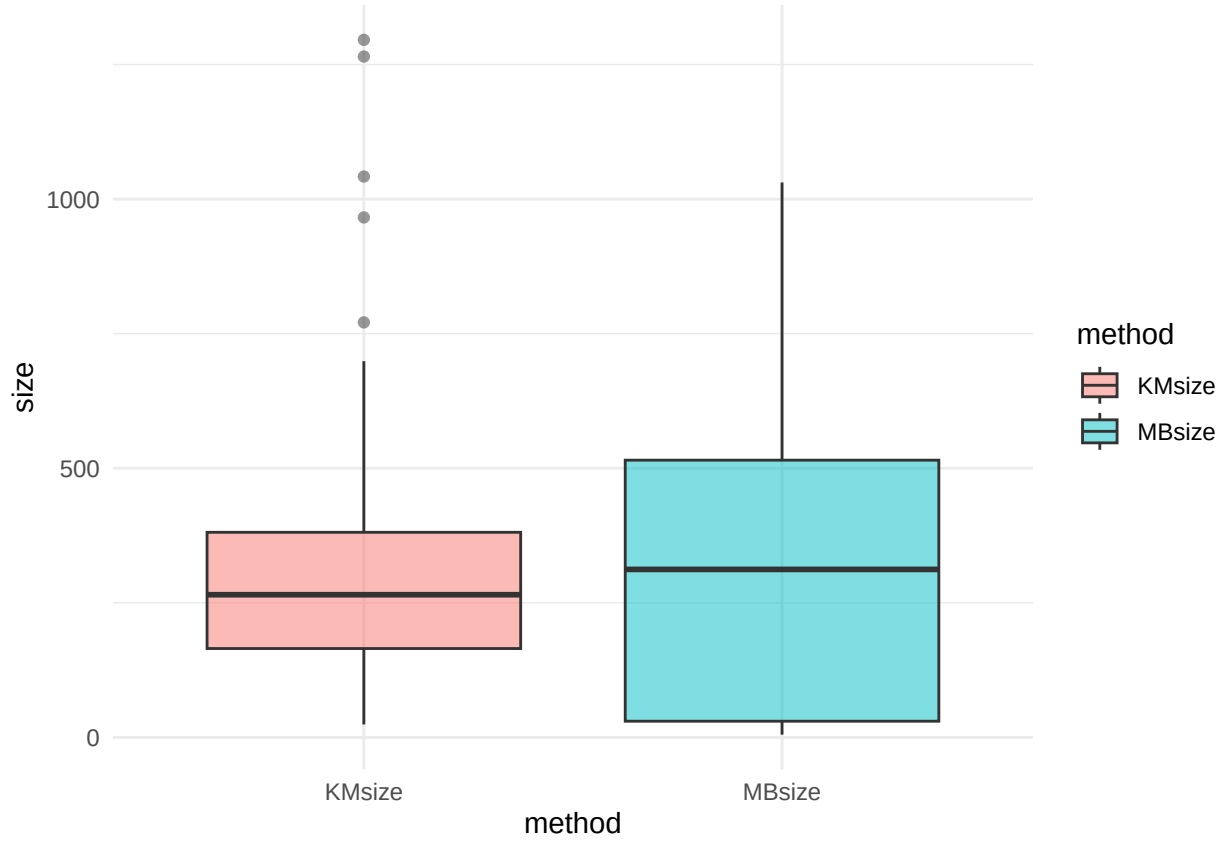


4 Comparison of the methods

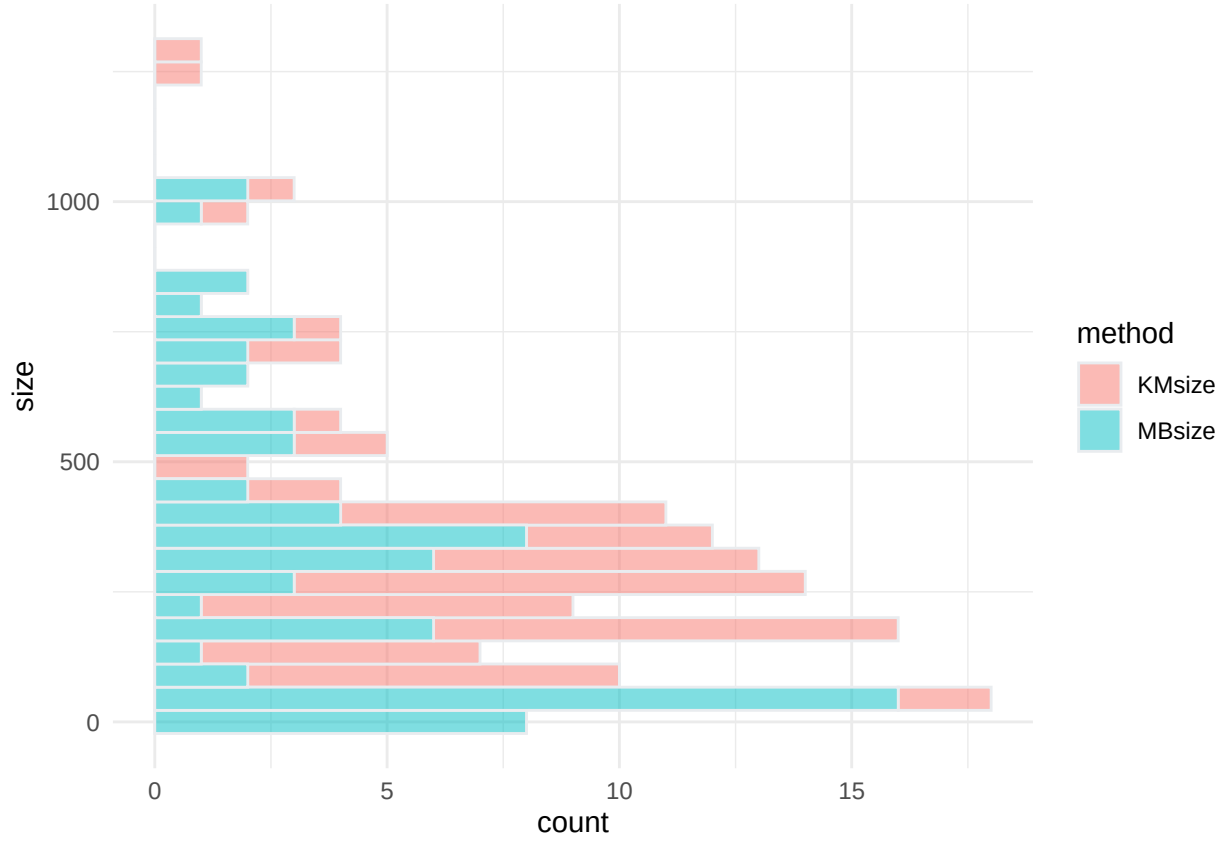
In orther to compare both used methods in this project, the table below shows some numeric data of the distribution of the clusters for each method. The number of clusters was fixed to 77, so this value and the average of cluster size are equal in both methods. However, with the standard deviation (SD) we can see that the Kmeans method has more clusters close to the size of the mean than the Model Based Approach. The Kmeans method also generates generaly larger clusters: both the largest and the smallest cluster are larger than the ones for the Model Based Approach.

Value	KM.method	MB.method
Quant. of clusters	77.0000	77.0000
Average of cluster size	320.2987	320.2987
SD of cluster size	247.6519	284.7911
Size of largest	1296.0000	1031.0000
Size of smallest	24.0000	5.0000

The boxplot below shows the distribution of the clusters sizes per method:



To finish with, the following histogram also shows the distribution of the clusters sizes per method:



5 Discussion and future works

Although the results of the method show that it is possible to identify and classify the position of solar protections, some clusters as desired, but they are clustered by similarity in other ways, such as shadows or intensity of colors. The clustering process could be optimized with the following recommendations:

- The success of the image recognition process is strongly related with the quality of the source images. In this project, images were taken 24/7, which means that they have different levels, contrast and colors. At the beginning of this project, I accounted for this possibility and I included a first step in the method to achieve uniform levels for images. It is performed using automatic tools provided by the `magick` package in R. However, this normalization tools does not seem to be enough. For future works, it is recommended to invest on a good normalization of images before working with them.
- Also the crop of the windows could be improved. In this project, the crop is made by a regular grid assuming that the façade is perfectly flat. However, the curvature of the lens makes some distortion and some windows do not get cropped as expected. For future works, it is recommended to correct the lens distortion and ensure that the windows are perfectly fitted with the crop.
- The last but not the least, the simplification of pixel darkness by row was used in this project to simplify the processing of the algorithms. However, it was not verified whether it has any counter effect on the results. For future works, this assumption should be verified.

6 Conclusion

This project is about the identification of the position of some manually controlled solar protections from images. To do so, I used two machine-learning algorithms to clustering the images by similarity, as this is an untagged learning process. Images are initially processed to enhance colors and contrast, then the darkness of the pixel rows are converted into numberic values to become the features for the final machine learning clustering process. The Kmeans algorithm and the Model Based Approach are used.

In conclusion, the Kmeans algorithm gets clusters of larger size, while the the Model Based Approach generate slightly better equally distributed clusters. However, the performance of a machine learning algorithm should not be evaluated from the size of the clusters, but from the quality classifying images according to the aim. From a qualitative analysis of the samples showed in the results section, the Kmeans algorithm groups together images that are similar in terms of position of the solar protection. The Model Based Approach, however, has some mixed results.

7 References and tools

Broman K (2022). *broman: Karl Broman's R Code*. R package version 0.80, https://CRAN.R-project.org/package=broman.

Hennig C (2023). *fpc: Flexible Procedures for Clustering*. R package version 2.2-10, https://CRAN.R-project.org/package=fpc.

Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M., Hornik, K.(2022). cluster: Cluster Analysis Basics and Extensions. R package version 2.1.4.

Ooms J (2021). *magick: Advanced Graphics and Image-Processing in R*. R package version 2.7.3, https://CRAN.R-project.org/package=magick.

Ochi S (2021). *magickGUI: GUI Tools for Interactive Image Processing with 'magick'*. R package version 1.3.0, https://CRAN.R-project.org/package=magickGUI.

Scrucca L., Fop M., Murphy T. B. and Raftery A. E. (2016) mclust 5: clustering, classification and density estimation using Gaussian finite mixture models The R Journal 8/1, pp. 289-317

R Core Team (2022). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.