LELEC2885 - Image processing and computer vision

Groupe 1 : Justin Loroy, Fabian Souris, Sebastien Tack

Introduction

We are asked to classify pixels of an image into category regarding the object it represents.
We divided it in three phases. We first created the features assigned to each pixels. We learned on it to make a decision model. We finally evaluated the performance of the model.

Features

Our features are extracted from both the pixels and the segments of the images. The segments are obtained thanks to the EDISON implementation of the mean shift segmentation algorithm.
All of these feature are normed between 0 and 1 for machine learning purpose, especially when using a SVM.

From the pixels

- **Color :** The Luv value of the pixel. We prefered this representation over the RGB representation because the luminance vary from an image to another and this allow our machine learning algorithm to take that into account.
- **Position :** the (x,y) position of the pixel. Can differentiate the sky from the pedestrian by example.

From the segments

- **Segment Size :** The size (in pixel) of the segment.
- **Edge Density :** The ratio between the number of pixels that belong to an edge into a segment and its size. The edge are computed by the MATLAB function *edge*, an implementation of the Sobel operator.
- **Mode Color :** The color with the highest probability inside a segment (in Luv).

Fusion of both

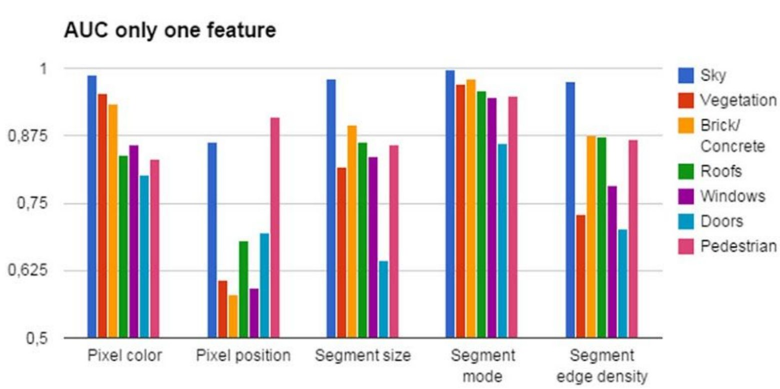
We infer the segment's features to the pixels, regarding which segment it belongs to. This method improve the performance, because it combine the best of the two worlds. Allowing us to bring information to the pixels about his neighbourhood.

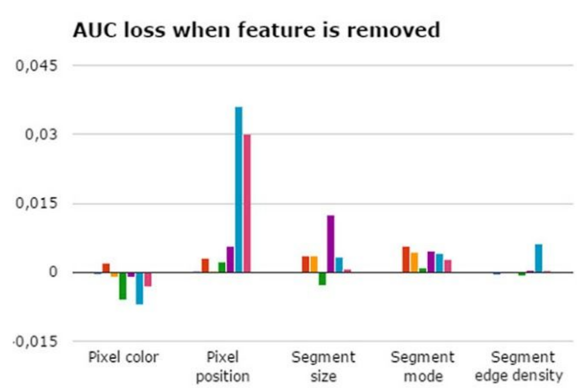
Feature significance / selection

We performed two tests to asses the significance of the 5 differents features.

- The AUC obtained with only one feature, to find the feature carrying the most information by it's own.
- The AUC with all but one feature, to find the feature giving the fewer extra information.

We used the random forest algorithm to assess the performances.





We can see that the segment mode is the feature bringing the most information on his own and, unexpectedly adding the pixel color feature reduce the performance indicator of our model. At this point we decided to remove this feature.

Performance indicators

Confusion Matrix

We extract some evaluation criterion from the data given by the confusion matrix

tn	tn	tn	fn	tn	tn	tn
tn	tn	tn	fn	tn	tn	tn
tn	tn	tn	fn	tn	tn	tn
fp	fp	fp	tp	fp	fp	fp
tn	tn	tn	fn	tn	tn	tn
tn	tn	tn	fn	tn	tn	tn
tn	tn	tn	fn	tn	tn	tn

- **Accuracy :** $\frac{trace(confusion\ matrix)}{sum(confusion\ matrix)}$
  Simplest way to visualize the performance (used for the grid searches)
- **Precision :** $\frac{tp}{tp+fp}$
- **Recall :** $\frac{tp}{tp+fn}$
  Both precision and recall allow us to have an illustration of which classes are rightly detected
- **Specificity :** $\frac{tn}{tn+fp}$
  Useful to have an illustration of which classes are rightly avoided

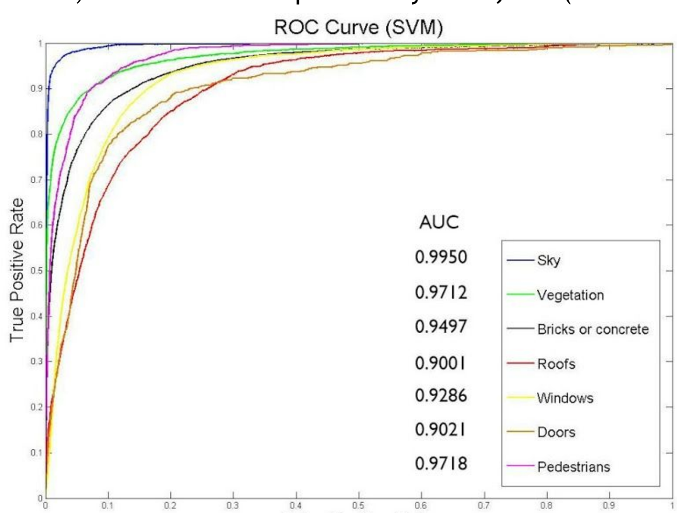
Machine Learning

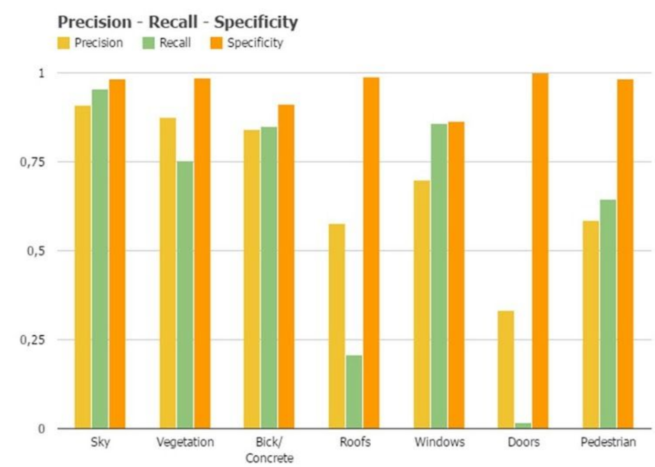
Process

First of all, we separated one tenth of the images to create a validation set that will be used to evaluate the final solution in real world conditions. We didn't do anything on this set during the whole development to avoid any overfitting on the data with our methodology.
To compare the algorithms and parameters, we choose to evaluate them with a 10-fold validation. Thus, we had always 90% of training set and 10% of test set of the 90% remaining data.

SVM

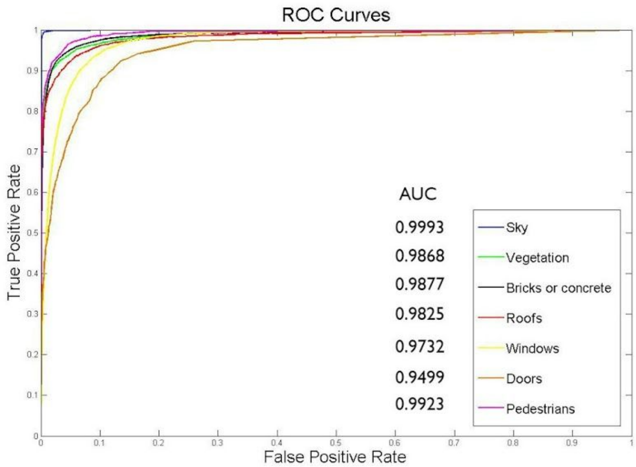
We used the libsvm implementation of the Support Vector Machine algorithm. It allow the model to make non-linear decision boundaries that instinctively makes it more precise on the categorization of the pixels. After the grid search, we obtained a top accuracy of **79,15%** (with the 5 features).





Random Forest

We tried another machine learning algorithm to confirm our instinctive choice for the SVM. But we found out that the random forest (we used the MATLAB implementation TreeBagger) gave better results than the SVM. After the grid search, we obtained a top accuracy of **90,27%** (without pixel color).



Grid search

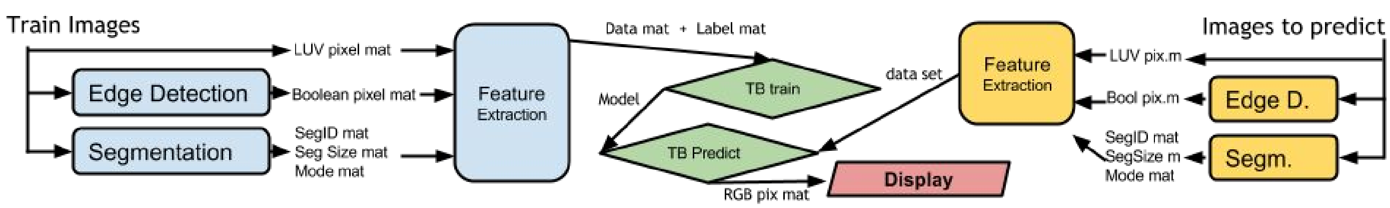
To improve the accuracy of the model, we adapted the parameters of the chosen machine learning algorithm using a grid search.

- SVM : On the parameters C, gamma and the kernel (radial kernel was far better than the other available ones).
- Random forest : On the number of generated random trees. We came to the conclusion that increasing this number would improve the performance until the gain become insignificant (~100 trees).

We also performed a grid search to improve the generation of meaningful segments. We used a fixed spatial bandwidth because we found it has a limited impact on the performances and increasing it increased hugely the computation time. We focussed our search on the parameters "range bandwidth" and "minimum region area", found to have the biggest impact on the performances in our use case.

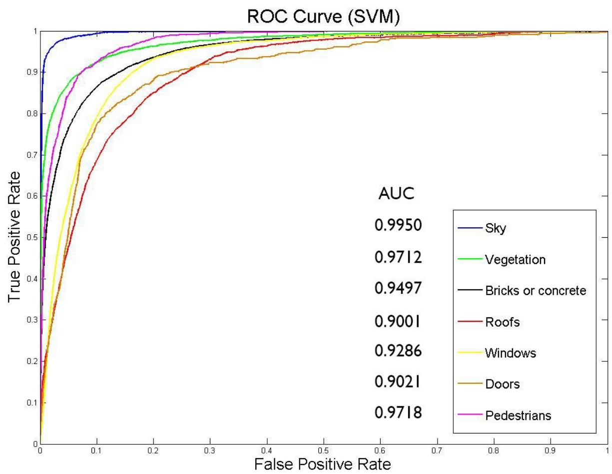
This grid search allowed us to improve the performance of our model significatively (around 1.2% gain in accuracy).

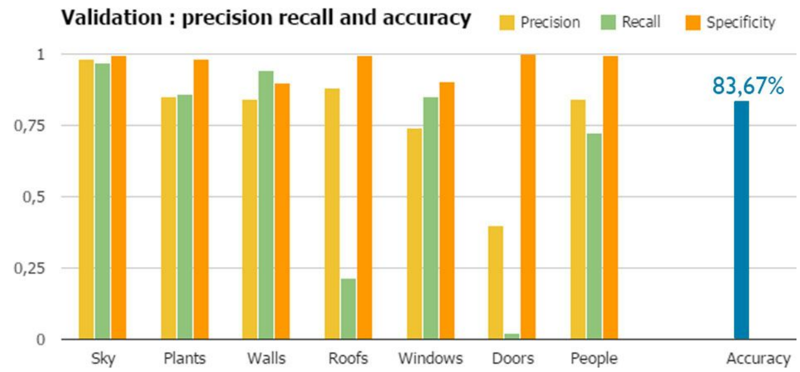
Code structure



Result

We are now considering the real performances of our algorithm on the test set, the 10% of untouched data.





Pixel color

It gives us bad results because of the possible local variation unlike the segment mode

Doors

These bad results can be explained because of the shape and size of this class. Close to the window one.

Pedestrian

The algorithm take this class as a trash class for the whole bottom of the image.

Possible improvements

SIFT

Since SIFT descriptor will provide information of the entourage of a pixel, giving the mean gradient, we thought that is could be useful but we did not implement it because the basic sift produce those information for the key points only and we wanted it to do it for each pixel. We tried to implement our own DoG matrix but we decided not to add it to the feature vector because we had already done all the grid search that are very time-consuming.

Texture detection

A useful feature would have been to implement a texture detector. It would have given information about the presence of Bricks or Roofing.

Gabor

We tested the Gabor method to detect the textures. But we did not add it to the features neither because it gave too much data to add to the feature vector.

Edge

We thought of using the edge density to detect possible texture. But after some trials, we discarded it because it did not respect the principle of luminance-invariance.

Bigger grid search

We have done a low resolution grid search because of lack of time and computation power. For example, we restrained the one for the segmentation to the two most important parameters instead of tuning the three of them.

Conclusion

We are satisfied about our 90% accuracy for the 10-fold validation, but we knew that for the validation this number would decrease a little bit. However we did not expect a 7% drop.
With more time, we would have implemented some additional features. After several failed attempts, the features we kept for the final solution are quite basics. We think it is possible to improve the accuracy with some other descriptors.
Regarding the grid search, we have not had enough time and computation power to make it as big as we wanted to.
Finally, we want to say that this project was really interesting and challenging.

Thank you for the organisation. And thank you for reading.

References

SVM	http://www.csie.ntu.edu.tw/~cjlin/libsvm/
EDISON	http://www.mathworks.com/matlabcentral/fileexchange/19406-3d-stereo-disparity/content/lankton_stereo/msseg/edison_wrapper.m
Mean Shift	https://courses.csail.mit.edu/6.869/handouts/PAMIMeanshift.pdf
DOC Matlab	http://nl.mathworks.com/help/matlab/