Automatic image annotation

Cristina Luengo Agulló Javier Navarro González Joao Rocha e Melo

Image Processing and Computer Vision, Group 8
École Politecnique de Louvain, Université Catholique de Louvain

Abstract

This paper talks about a categorizing system for pictures with several elements. The final goal is to be able to identify if a pixel belongs to one of seven categories: Sky, bricks, vegetation, pedestrians, windows, roofs and doors. In the begining, users were asked to manually identify a large amount of pixels as certain regions. Features were then retrieved from pixels to classify the regions, making it possible to identify other pixels. This way, it is possible to determine to which region a pixel belongs. In the overall, this process proved to be very efficient in identifying flat features like the sky, but not so efficient in the identification of features like doors and pedestrians

1 Introduction

The main objective of this project is to be able to identify different areas in images containing elements of interest (e.g. sky, vegetation, etc.). To do so, the pipeline used involves RGB image transformation to a gray scale[1], filtering, watershed segmentation, feature extraction and classification (carried out in two steps: train and validation, and test). Finally, performance assessment allows us to draw conclusions about the method used. The pipeline described can be found in figure 1

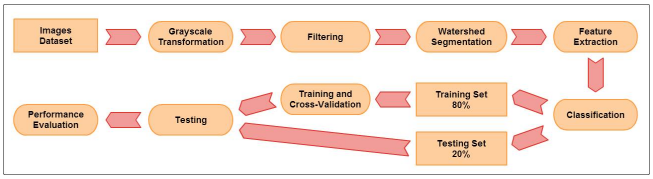

Figure 1: Pipeline of the system

2 Image segmentation

Features are extracted using the position of the pixel. For each pixel, the three color components (RGB) are obtained for the color analysis. However, for all the other features, a transformation of the image to grayscale is used, so in this case, the input is a one channel image instead of a three channel one.

2.1 Watershed method and Filtering

Watershed method is a segmentation tool based on the gradient. To use this segmentation method, Matlab has a function that generates a watershed map [1]. This function is a threshold based method and creates too much oversegmentation. Although the best way to solve this problem is to change the threshold value, due to privacy reasons, this is not possible in Matlab. To solve this problem, the images were filtered using a median filter with a 3-by-3 neighboorhood in order to reduce the amount of regions.
In this project, the watershed has as input a grayscale image and builds a mountain map with the gradient feature. After this, it considers that pixels are in the same region if they belong to the same "valley" of the mountain map.
This method was chosen from many based mainly on the fast computational time it takes. Since there was no way to compare the results at this point of the project, no other criteria could be used.

3 Feature extraction

3.1 Colour

The colour space used in this project was RGB. We chose this space because we wanted to see the impact of the color in relation with the human eye perception, and in Lab space for example, many of the colors fall outside the gammut of human vision, and are therefore purely imaginary.
The approach used was not a direct one, so instead of retrieving the RGB values from each pixel, an average was calculated. Analysing all the RGB values from all the pixels in the same segmented region as the manually annotated pixel, it is possible to compute the average of each colour component and it is this average that is going to be the input of the classifier. Although this algorithm is computationally more expensive because it computes all the pixels of the image and not just the ones from the manual anotation, this approach was chosen to increase the robustness of the process. That way, if the annotated pixel was for some reason very different from its surroundings, this method is able to correct that mistake.

[1]Conversion from RGB to Grayscale is simply a weighted sum of the R,G and B componets where $I = 0.3R + 0.59G + 0.11B$

3.2 Gradient

The gradient used is the pixels difference in a grayscale image. After transforming the image to grayscale, the gradient is computed simply by subtracting the values from the surrounding pixels.This is an interesting feature in this particular project because going through the regions, one could expect the gradient in regions like the sky to be very low and in the bricks, vegetation and pedestrians to be very high. In doors, windows and roofs, the gradient can be very low or very high, it depends on the image.

3.3 Texture

Texture extraction is computed using a range filter. Each pixel value in the feature matrix is calculated by applying a 3x3 window to each pixel of the input image. The value of an output pixel will be the subtraction of the lowest value from the neighbors contained in the window of the input image, from the highest value. This method allows us to compute a local range for each pixel, since texture values can not be calculated isolated from other pixels.

3.4 Number of pixels of the regions

The number of pixels of a region is simply the amount of pixels in the watershed region of the selected pixel. This feature was chosen because of the segmentation method. Since the watershed method uses gradient based segmentation, in regions where the gradient is low, there will be a lot of pixels in the same region, therefore a big region size will be common to pixels with low gradient and a small region size to pixels with big gradient.

3.5 Position of the regions

For each pixel of the image it is retrieved its region from the label matrix that the watershed function builds. With all the points of each region, its (x,y) average position is computed, which is assigned to all of the points from the same region. This feature will give us useful information about the points, because the classes usually are in the same sections on the images (e.g. sky is usually on the upper side of the pictures, pedestrians on the lower side, etc.)

4 Classification

Once image segmentation and feature extraction from all the manually annotated pixels has been performed, a classifier is used to automatically annotate pixels. The classifying task is carried out with the use of the statistical tool Weka[2], and it is done in two main steps: train and test.
Therefore, we use a large dataset comprised of all the manually annotated images from all the groups that joined course, and we divide it randomly in two scrambled subsets: 80% of the images for the training phase (561), and 20% of the images for testing (140).
The train dataset is first fed to the Weka tool, and the classifier to use is selected along with the different parameters it needs.
The classifier chosen implements the K-Nearest Neighbor algorithm [2] [3] (called *Ibk* in Weka). This classifier was chosen due to its simplicity and intuitivity. Given a set of classified pixels, when a new pixel needs to be classified the algorithm looks for the k nearest neighbors in the feature space (according to their euclidean distance), and assigns to that pixel the most frequent class of its neighbors. Since some neighbors might be more distant than others, weighting is applied in order to allow closer pixels to influence more in the final class decision (count as more votes).
That way, the weighting method and the number of neighbors are parameters to the algorithm that need to be tuned in order to achieve the best performance results.
To assess performance when tuning the parameters, an additional step is carried out: validation. The train set is divided in two subsets: 66% train and 34% validation, and cross-validation is applied in 10 folds (each iteration with a different pair of subsets). These last ones are parameters for the Weka tool itself that need to be tuned before running the classifier.
Once both the classifier and Weka's parameters are set, evaluation metrics for runs with different parameters values are compared in order to see which parameter values for the classifier give better results.

4.1 Train and validation

Our goal during the validation phase is to maximize the smallest value of the area under the ROC curve (AUC) for all the classes. Since Weka computes the average of the ROC areas using the number of instances in each class as a weight (weighted average) and the dataset is very unbalanced (figure 4), we consider that this value will not be reliable, and therefore, we choose the smallest AUC value as our target to maximize.
That way, different runs with different values for the number of neighbors and the distance weighting method are carried out.

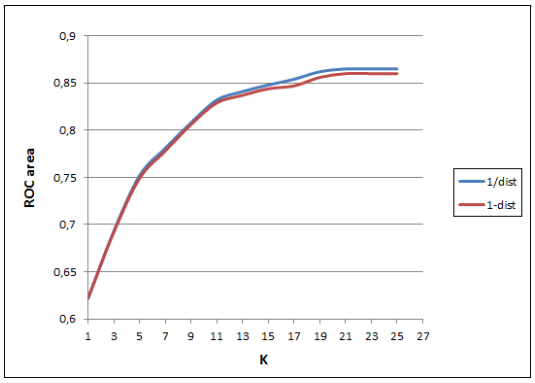

Figure 2: Graphic showing ROC areas for different classifier parameter values

[2]Weka is a collection of machine learning algorithms for data mining tasks. It contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. Available from: http://www.cs.waikato.ac.nz/ml/weka/

The results show that the highest ROC area (or AUC) is achieved when using

- K = 21[3].
- Distance weight = 1/distance.

Therefore, these will be the parameters values used during the testing phase, since they proved to achieve better performance results according to our criteria.

4.2 Test

Once parameters have been set, the classifier is fed with the test set (comprises samples not seen in the train set) over the train model. The different performance values obtained for the test set are shown in figure 3.

```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.94      0.016      0.917      0.94      0.928      0.994    sky
                 0.571     0.015      0.826      0.571     0.675      0.941    veg
                 0.874     0.153      0.741      0.874     0.802      0.937    bricks
                 0.244     0.018      0.54       0.244     0.336      0.878    roof
                 0.75      0.136      0.672      0.75      0.709      0.898    window
                 0.034     0.001      0.417      0.034     0.064      0.851    door
                 0.615     0.012      0.58       0.615     0.597      0.961    peds
Weighted Avg.    0.743     0.094      0.733      0.743     0.725      0.93

=== Confusion Matrix ===

    a    b    c    d    e    f    g   <-- classified as
 1129    0   29    2   41    0    0 |   a = sky
    7  497  184   10  154    1   18 |   b = veg
   19   21 2223   40  196    0   44 |   c = bricks
   17    8  191  150  250    0    0 |   d = roof
   59   62  296   76 1552    6   19 |   e = window
    0    7   25    0  100    5    8 |   f = door
    0    7   54    0   16    0  123 |   g = peds
```

Figure 3: Output performance results and confussion matrix from the Weka tool

4.3 Performance interpretation

Since the dataset used is unbalanced, accuracy is not a reliable metric to consider, so we will assess performance by observing the values of the F-measure and the AUC (Area under a ROC curve), as well as the confussion matrix:

- F - measure: Harmonic mean of Recall[4] and Precision[5]
- AUC [4]: The ROC curve plots the False positive rate (X axis) of a classification against the True positive rate (Y axis). In the Weka tool, ROC curves are given for each class, considering a one-against-all scenario (binary classification that assigns a positive label to the class of interest and a negative one to all the other classes).
  The curves are generated by sorting the predictions produced by the classifier in descending order of the probability it assigns to the positive class. In the case of the KNN classifier, the probability distribution is computed by assigning a probability of 1 to the class of the majority of the neighbors and a 0 to the others. Since the tool knows the actual class labels in the historical data, it can step through this list and compute the number of TP, TN, etc. at each point in the list (each step in the list becomes one point on the curve). Each point in the list also corresponds to setting a threshold on the probability assigned to the positive class. Therefore, ROC curves show the trade-off between positive cases being either over-classified or miss-classified.
  The AUC is the area found under the ROC curve, and it's calculated using the Mann Whitney statistic [5].
- Confussion matrix: We assume the following:
  - True Positives (TP): Pixels classified as a certain class and truly belonged to that class.
  - True Negatives (TN): Pixels not classified as a certain class and did not belong to that class
  - False positives (FP): Pixels classified as a certain class but did not belong to that class
  - False negatives (FN): Pixels not classified as a certain class but, in fact, belonged to that class

Our main expectation for the classification task was to identify the sky very well. Since it usually is a smooth area, gradient and texture values will be low, the number of pixels in the segmented regions contained will be larger than in other areas (because of the watershed method, given the sky's low gradient), and the centroid position of the segmented regions will be usually at the top of the images.
Therefore, since most of the chosen features can help distinguish flat areas from textured ones, the classifier should be able to make good predictions about the sky. Furthermore, we would expect windows to be the most missclassified area, since they can reflect their environment and therefore could be confused with the sections of the image they are reflecting. Also, we would expect high gradient and texture values, and small numbers of pixels on the segmented regions for vegetation, bricks and pedestrians, so the classifier might confuse those areas when their values for the other features are similar (e.g. vegetation close to pedestrians or bricks, similar colour for vegetation and bricks, etc).
Moreover, windows and doors might be confused with each other too since their features values can be similar when they are both made of the same kind of material (e.g. glass doors). And since windows can usually be at any place in the image, the centroids of the segmented regions might not help distinguish both areas.

4.4 Results

Observing the results given in figure 3, we can draw different conclusions.
On one hand, the sky is indeed very well classified, since both the F-measure and AUC have high values and most of the classified instances are true positives. The instances missclassified as windows may be due to the fact that they reflect the sky or have a flat surface.

[3] Uneven values are chosen to avoid ties when voting

[4] $Recall = \frac{TP}{TP+FP}$

[5] $Precision = \frac{TP}{TP+FN}$

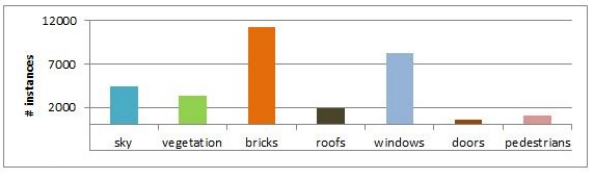

Figure 4: Graphic showing the number of instances per class

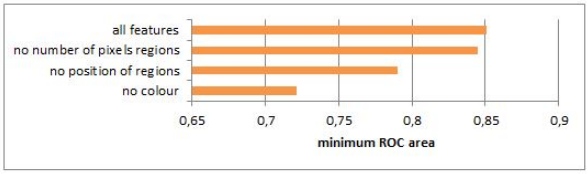

Figure 5: Graphic showing minimum ROC areas when removing different features from the dataset

On the other hand, the worst classified are doors, since they are mostly confused with windows. This does not completely meet our assumptions, since we expected doors to be confused with windows and the other way around, but not many windows are classified badly as doors. This could be due to the fact that the number of instances for doors is significantly lower than the number of instances for windows in the dataset, so there will be less points referring to doors in the feature space.
Therefore, when a new pixel needs to be classified, since doors and windows might have similar features values their points might be close in the feature space. Nevertheless, since there are a lot more points for windows than for doors, the classifier would find as the nearest neighbor of a new pixel a window point most of the times.
Another remarkable observation is the fact that roofs are missclassified as bricks and windows three times more than they are well classified. This could be due to the fact that a lot of images have windows on the roofs, so the position of the segmented images might influence in the missclassification, and since roofs have similar characteristics as bricks (e.g.high gradient, close to each other), given the chosen features they can be confused as well.
In order to see the impact of features in performance, some tests were carried out removing the feature of interest everytime (only relevant tests are shown), and performance measures were compared with the ones obtained when using all the features.

4.4.1 Without Colour

Removing the colour feature causes an overall decrease in performance and impacts it the most (as can be seen in figure 5) . Sky is still well classified because features like gradient, texture, position of the region and the number of pixels of the segmented regions help identify it, but since vegetation and bricks might have similar values for the remanining features (e.g. high gradient and texture values, low number of pixels in the segmented regions, they are sometimes close in the image), they are more confused with each other.

4.4.2 Without Positions of the regions

Testing the data without using the position feature led to some missclassifications. Mainly, the impact of not using the position was the classifcation of pedestrians as bricks and the absence of doors well classified. This happens because features like pedestrians, vegetations, bricks and doors are very high gradient surfaces and with a small segmented region size associated. Therefore, when the position is retreived from the classificaton process, the doors and pedestrians (that usually are found at the bootom of the picture) loose very important information and are therefore missclassified.

4.4.3 Without Gradient

To test this model without the gradient feature would lead to an inconclusive test. This is one of the downsides of this process. Due to the Watershed segmentation method, the gradient is already influencing the entire project (for example, high gradients will lead to small segmented region sizes). One of the ways to solve this problem could be to use another segmentation method. However this solution would be too expensive in terms of time, regarding the fact that each computation for the whole data takes a rather large amount of time.

4.4.4 Without Number of pixels of the regions

Removing the number of pixels of the regions feature does not give us much relevant information. The class that suffers it more is the pedestrians, losing almost half of its well classified pixels between roofs and windows. This missclassifications are because pedestrians are the most variable class. The clothes of the pedestrians are of many differents colors and their skin as well, and without this feature we can not discrimine this different colors as good as with it, because the segmented regions of pedestrians are smaller than in roofs and windows.

4.5 Conclusions

After testing and result analysing, it can be concluded that some performances in this algorithm are not the best, as described in section 4.4. The two biggest changes made to improve this method could be: 1) Other segmentation method or 2) The choice of other features. Another segmentation method not based on the gradient would lead to a change in the results. As seen above, some features made this algorithm robust when classifing certain regions, which indicates that adding/replacing features can lead to a significant improvement of the results.
Finally, the change of the classification algorithm could also lead to better results, since in the presence of class imbalance, a query instance will often be classified as belonging to the majority class and as a result many positive (minority class) instances will be misclassified.

References

[1] S. Beucher, *The Watershed Transformation applied to Image Segmentation,* Centre de Morphologie Mathématique, Fontainebleau Cedex, France

[2] Gondgde Guo, Hui Wang, David Bell, Yaxin Bi and Kieran Greer *KNN ModelBased Approach in Classification* School of Computing and Mathematics, University of Ulster, Northern Ireland, Uk and School of Computer Science, Queen's University Belfast, INN, Unk

[3] Zdravko Markov and Ingrid Russell *An Introduction to the WEKA Data Mining System,* Available from: http://www.cs.ccsu.edu/~markov/weka-tutorial.pdf

[4] Tom Fawcett *An introduction to ROC analysis,* Institute for the Study of Learning and Expertise, 2164 Staunton Court, Palo Alto, CA 94306, USA

[5] David Faraggi and Benjamin Reiser *Estimation of the area under the ROC curve,* Department of Statistics, University of Haifa, Haifa 31905, Israel

[6] David W. Aha, Dennis Kibler and Marc K. Alber, *Instance-Based Learning Algorithms,* Information and Computer Science Department, University of California, Irvine, CA 9271, USA

[7] D. G. Bailey and R. M. Hodgson, *Range filters: local-intensity subrange filters and their properties,* University of Canterbury, Private Bag, Christchurch, New Zealand

[8] Mihran Tuceryan and Anil K. Jain, *Texture Analysis,* Computer Science Department, Michigan State University, East Lansing, MI 48824-1027, USA