Unsupervised classification and clustering of image features for vehicle detection in large scale aerial images

Daniel A. Lavigne
Spectral and Geospatial Exploitation Section
Defence Research and Development Canada
Quebec, Canada
daniel.lavigne@drdc-rddc.gc.ca

Samir Sahli, Yueh Ouyang, Yunlong Sheng
Center of Optics, Photonics, and Lasers
Laval University
Quebec, Canada
yunlong.sheng@copl.ulaval.ca

Abstract – *This paper presents a set of algorithms for vehicle detection in large scale aerial images. Vehicles are detected based on geometric and radiometric features, extracted within a multiresolution linear Gaussian scale-space. The image features, described by their local structures, are classified using support vector machines. Classified features are then clustered by an unsupervised affine propagation clustering algorithm, within a feature-level fusion scheme. Subcomponent of vehicles' body parts are aggregate together with respect to shared spatial relations and based on constraints on the orientation of detected vehicles. Experimental results using large scale aerial imagery demonstrate the efficient and robustness of the proposed algorithms for the detection of vehicles in an urban environment.*

Keywords: Vehicle detection, local image features, unsupervised classification, SVMs, clustering.

1 Introduction

Large scale imagery acquired by remotely sensed airborne and spaceborne sensors has important civilian and military uses. Such imagery is valuable for a broad range of field applications, including: the extraction of geographical data for geoinformation systems, traffic control management, detection and identification of targets, urban planning, vehicle counting in the context of military reconnaissance, change detection, 3D urban landscape modeling, etc. Amid these applications, the detection of vehicles in large scale imagery acquired by high resolution sensors has received great attention over the years in both the remote sensing and computer vision communities.

However, detecting automatically such inconsistent instances in multisensor and multitemporal datasets is a difficult task since the set of generic features (i.e. the vehicle's main body parts such as the rooftop, the hood, and the windshield) can be acquired by different sensors, at different time, and according to various viewpoints. Accordingly, the image features related to the vehicles could be represented in the image content with different spatial and spectral resolutions and intensity levels. Moreover, some vehicles could heavily interfere with the environment, producing occlusions and shadow areas within the scene. Therefore, many image processing algorithms have been proposed over the years in order to cope with this challenging problem. For instance, vehicles can be considered as explicit 2D instances to be detected with edge masks by extracting the four sides of the rectangular boundary [1], or modeled with rectangles of different sizes [2]. Other algorithms considered vehicle detection as an explicit 3D object recognition problem to accommodate the change of viewpoint and make use of the shadow cues. Such algorithms have been proposed using a Bayesian network integrating all image features [3], or based on a hierarchical 3D model approach [4]. Additionally, algorithms considering an implicit model of the vehicles to detect have also been developed [5] [6].

Following the work achieved by such valuable algorithms, this paper presents a set of algorithms to detect vehicles in large scale aerial images. The algorithms detect geometric and radiometric features extracted from the vehicles within a multiresolution linear Gaussian scale-space. Then, the image features described by their local structures are classified using support vector machines (SVMs). Extracted features are subsequently clustered by an unsupervised affine propagation algorithm within a feature-level fusion scheme. Unfolding vehicles' body parts are aggregate together according to their shared spatial relations and based on constraints on the orientation of the vehicles. Experimental results using large scale aerial images demonstrate the efficient and the robustness of the proposed algorithms for the detection and counting of vehicles in an urban environment.

This paper is organized as follows. Section 2 describes the multiresolution feature detection and classification algorithms used, while section 3 depicts the unsupervised affine propagation clustering approach. Section 4 presents the experimental results following the use of the image fusion scheme on large scale aerial images. Finally, section 5 concludes the paper and suggests ways to improve further the algorithms developed.

2 Image feature detection

In order to cope with the wide range of vehicle types that potentially can occur within typical aerial imagery, the image feature detection algorithms to devise must be able to discriminate major vehicle's substructures, namely: the windshield, the rooftop, and the hood. Figure 1 illustrates an aerial image of an urban scenario with spatial resolution of 11.2 cm per pixel and a closeup view of some vehicles substructure components.

(a) Coarse view (b) Closeup view

Figure 1: Aerial imagery of an urban scenario (spatial resolution of 11.2 cm per pixel). 1(a) Coarse view of the large scale aerial image and 1(b) Closeup view of some vehicles components.

The following subsections describe the multiresolution framework used to extract the vehicle main components, followed by their classification using support vector machines.

2.1 Multiresolution feature detection

Within a multiresolution linear Gaussian scale-space representation, local image features robust to geometric distortion, illumination, noise, and image scale are detected, using a scale-invariant feature transform [7]. This method has been chosen since a comparative performance evaluation study of different local descriptors showed its superiority over a broad range of detectors [8]. The multiscale representation used is a special type of scale-space representation that comprises a continuous scale parameter and preserves the same spatial sampling of the image at all scales [9]. Let $f : R^N \to R$ represent any given signal. Then, the scale-space representation $L : R^N \times R_+ \to R$ is defined by letting the scale-space representation at zero scale be equal to the original signal $I(x, y; 0) = f$, and for $t > 0$:

$$L(x, y; t) = g(x, y; t) * f \tag{1}$$

where $t \in R_+$ is the scale parameter, and $g : R^N \times R_+ \backslash \{0\} \to R$ is the Gaussian kernel in arbitrary dimensions N:

$$g(x; t) = \frac{1}{(2\pi t)^{\frac{N}{2}}} e^{-\frac{x^t x}{2t}} \tag{2}$$

$$= \frac{1}{(2\pi t)^{\frac{N}{2}}} e^{-\sum_{i=1}^{N} \frac{x_i^2}{2t}} \quad (x \in R^n, x_i \in R) \tag{3}$$

The scale-space family L can equivalently be defined as the solution of the diffuse equation (with initial condition $L(\cdot; 0) = f$):

$$\delta_t L = \frac{1}{2} \bigtriangledown^t \bigtriangledown L = \frac{1}{2} \sum_{i=1}^{N} \delta_{x_i^2} L \tag{4}$$

Multiscale spatial derivatives can then be defined from this scale-space representation:

$$L_{x^n}(\cdot; t) = \delta_{x^n} L(\cdot; t) = g_{x^n}(\cdot; t) * f \tag{5}$$

where g_{x^n} denotes a possibly missed derivative of some order n. In fact, n should be interpreted as a multi index $n = n(n_1, ..., n_N)^t \in Z^N$ where $n_t \in Z$. Thus, $\delta_{x^n} = \delta_{x_1^{n_1}}, ..., \delta_{x_N^{n_N}}$, where $x = (x_1, ..., x_N)^t \in R^N$ and $x_t \in R$. Local image features are detected as maxima at each scale of the Difference-of-Gaussians (DoG) occuring at multiple scales, generated from the scale-space representation as:

$$D_{OG}(x, y; t) = L(x, y; kt) - L(x, y; t) \tag{6}$$

where k is a constant factor separating two nearby scales t in the scale-space representation of equation 5, and while the characteristic local scale of each image feature is selected automatically on the basis of the maximum of the normalized Laplacian $|\sigma^2 L_{xx}(x, \sigma) + L_{yy}(x, \sigma)|$.

Local orientation of each image feature is determined by the identification of peaks in the histogram of the gradient orientations of detected features [12]. As such, a feature vector is used to form a descriptor of each image feature, where the length characterizes the pixels neighborhood of the image feature sampled at eight gradient orientations over a four-by-four array of histograms sampling the region of interest. Consequently, for each image feature, a 128-element feature vector is formed.

Extracted image features are then classified by means of their respective feature vectors as *vehicle* or *non-vehicle* instances using support vector machines. In current vehicle classification case scenario, the training set is defined by labeling some vehicle samples. Figure 2 illustrates vehicles labeled for the training set.

2.2 Classification using SVMs

Support vector machines [13] are used for classification of the extracted image features by construction of a set of hyperplanes in high dimensional feature space [14]. The algorithm tries to find a large margin linear classifier from a subset of learning samples labeled $+$ and $-$ lying in R^n, called *Support Vector (SV)*. This corresponds to a two-class problem so that $f(\mathbf{x}, \alpha) \in \{+1, -1\}\ \forall \mathbf{x}, \alpha$. For a learning set $A = \{(x_1, y_1), ..., (x_k, y_k)\}$ of k pairs where $x_i \in R^n$ and $y_i \in \{+1, -1\}$, the algorithm maps the x_i vectors in subspace $\mathbf{H}$ using a linear function $\phi : R^n \to \mathbf{H}$ in order to determine the optimal hyperplan $(\mathbf{w}, b)$ separating the two classes of $\mathbf{H}$. The new class y of sample x is then defined by $y = sign(\mathbf{w} \cdot \phi(x) + b)$. The hyperplan is considered optimal if it maximizes the distance between its most close neighbor samples. By choosing the kernel function $\psi(x_i, x_j) =< \phi(x_i), \phi(x_j) >$, the corresponding dual problem can be represented by the decision function:

$$f(x) = \sum_{x_i \, \epsilon \, \xi} \alpha_i y_i \psi(x_i, x) + b \tag{7}$$

where $\{\alpha_i\}$ and b correspond to the optimal solution of the dual problem and ξ is the subset learning samples forming the SV.

The vehicles will be classified by training the SVM with the feature vectors of the image features, in order to elaborate a $p-1$ dimensional hyperplane separating the image features according to a *two-class* classification problem: $[+1$ for *vehicle*, -1 for *non-vehicle* instances$]$. Feature vectors' elements are map into a feature-space in which they can be separated by a linear hyperplane, using the Euclidean distance between feature vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, by means of the radial basis function kernel:

$$\psi(\mathbf{x}_i, \mathbf{x}_j) = \phi^t(\mathbf{x}_i)\phi(\mathbf{x}_j) = e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2} \tag{8}$$

where γ is the hyper-parameter.

With the soft margin model [16], the SVM allows mislabeled examples during the training phase, yielding the following optimization problem to solve:

$$\min_{\mathbf{w},b,\xi} \left(\frac{1}{2}\mathbf{w}^t\mathbf{w} + C\sum_{i=1}^{l} \xi_i \right) \tag{9}$$

subject to the constraint:

$$y_i(\mathbf{w}^t\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i \tag{10}$$

where $\xi_i > 0$ is the slack variable which measures the degree of misclaaification, $C > 0$ is the penalty parameter of the classification errors, **w** is the support vector normal to the hyperplane, b is the offset of the hyperplane to the origin. Thus, the training of the SVM aims to solve the support vector **w** that maximize the margin of separation, which is proportional to $1/|\mathbf{w}|$ according to the geometrical interpretation. The hyper-parameters C and γ should also be determined by the training, which is achieved by cross-validation. Finally, the optimization problem defined in eq. 9 is solved using a quadratic programming technique [15]. Figure 3 illustrates some classification results.

3 Unsupervised affinity propagation clustering algorithm

Image features previously detected within the multiresolution scale-space and classified by the SVM as either belonging to the *vehicle* or *non-vehicle* classes, are

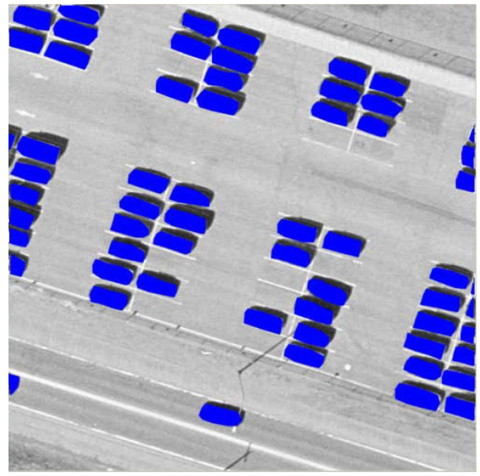


Figure 2: A training set of vehicles is defined.

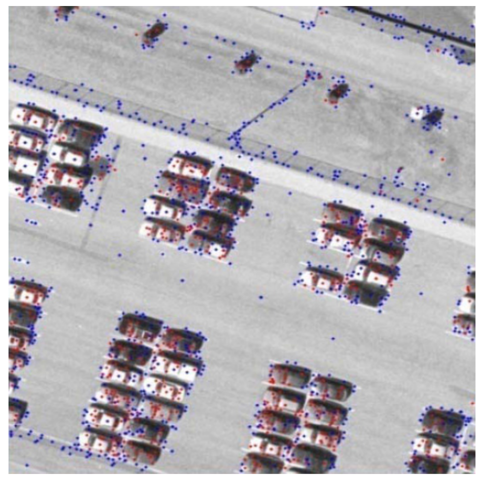


Figure 3: Classified image features. Red dots are image features correctly classified as belonging to the *vehicle class*, while blue dots are *non-vehicle* image features.

then clustered together according to the unsupervised affinity propagation clustering (UAPC) algorithm, which is based on the similarity between pairs of image features [11].

This is a mandatory step since some characteristics of subcomponents of a single vehicle produced many detected image features (such as on the rooftop, the windshield, and the hood (Fig. 4)). Therefore, the classification results must be improved by grouping images features into instances of features of the same vehicle's subclass.

A way to cope with that is to cluster the classified image features based on given measures of similarity. The UAPC algorithm takes as input measures of similarity between pairs of data points. Clustering data points x_i with $i = 1, ..., N$ consists of identifying a set of cluster centers **c** and assigning every x_i to a center c_i so that with

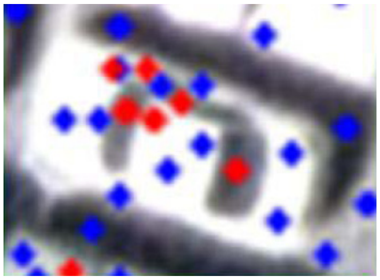


Figure 4: Detection of subcomponents of an instance of the *vehicle* class. Red dots are image features correctly classified as belonging to the *vehicle class*, while blue dots are *non-vehicle* image features.

the similarity between x_i and x_k, defined as a negative Euclidean distance $s(i,k) = -\|x_i - x_k\|^2$, the following fitness function is maximized:

$$S(\mathbf{c}) = \sum_{i=1}^{N} s(i, c_i) \tag{11}$$

Contrary to other conventional clustering algorithms, the UAPC algorithm considers all data points simultaneously as potential centers and iteratively exchanges the similarity messages between data points until a set of centers and corresponding clusters gradually emerge. The algorithm is based on this exchange of evidence at the bottom of the data population to induce a global partition.

The *responsability* $r(i,k)$, sent from data points i to candidate center point k, reflects the accumulated evidence of how well-suited point k deserves to serve as the center point for point i, taking into account other potential centers for point i. It is computed by:

$$r(i,k) \leftarrow s(i,k) - \max_{k' \neq k}\{a(i,k') + s(i,k')\} \tag{12}$$

following its initialization to zero $a(i,k) = 0$.

The *availability* $a(i,k)$ sent from candidate center point k to point i reflects the accumulated evidence of how appropriate it would be for point i to choose point k as its center, taking into account the support from other points that point k should be a center. It gathers evidence from data points as to whether each candidate center would make a good center:

$$a(i,k) \leftarrow \min\Big\{0, r(k,k) + \sum_{i' \notin \{i,k\}} \max\{0, r(i',k)\}\Big\} \tag{13}$$

The UACP algorithm takes into account a collection of real number $s(k,k)$ (i.e. self-similarity) for each data point k so that the ones with large values of $s(k,k)$ are more likely to be chosen as centers, i.e. the sum of squared errors between data points and their nearest centers is small. Then, real-valued messages exchanged between data points until a high-quality set of centers and corresponding clusters gradually emerges. Thus, after iterative update of all data points, k is assigned as a center for i such that the pair (i,k) generates a maximum of the sum of $r(i,k) + a(i,k)$ as $arg\max_k\{r(i,k) + a(i,k)\}$.

4 Experimental results

A set of aerial image of 1000×1000 pixels with resolution of 11.2cm per pixel was used as relevant dataset to test the proposed algorithms. Although the spatial resolution of the dataset can be considered as relatively high, it is still of low resolution to extract vehicles since such typical instances appear at this resolution within a rectangle of 40×20 pixel size (Fig. 1(b)). Therefore, only large subcomponent parts of vehicles (such as the trunk, hood, windshield, and associated shadow) can be distinguished at this resolution. Additionally, related image features to be extracted from the vehicles are distorted by induced noise, specular reflection, viewpoint variation, and low color contrast level.

4.1 Extraction of image features

Image features of vehicles are extracted using the linear multiscale gaussian-scale space approach described in subsection 2.1. Gradient-based features are mostly presented in the textured regions of the image where vehicles are located, and thus, absent in the plain regions where parking lots and roads are found. However, numerous image features related to buildings, lawn, bushes, and shrubs are also detected since they all contain structured and textured surfaces. The image features are described by a 128 feature vector elements representing the histograms of the gradient orientations computed within a local patch of size $12\sigma \times 12\sigma$.

4.2 Training the SVM with image features

Three aerial images in which there was a total of 89 vehicles were used to train the SVM. A total of 23923 image features was extracted from the training images at the first octave $i = 0$ of the linear multiscale gaussian scale-space representation. A hand made label map was created for each image by a careful selection of vehicle regions: $\{+1\}$ and $\{-1\}$ for *vehicle* and *non-vehicle* class features, respectively. Therefore, the training set integrates a total of 23923 labeled feature vectors of 128 elements. A 10-fold cross validation was applied in the training. The training feature vector set was divided into 10 equal-sized parts with one of the 10 parts kept out as the validation set and the remaining 9 parts combined for the training. The SVM has been trained using the training set with given hyper-parameter values C and γ from equations 8 and 9, and tested along the prediction accuracy of the validation set. The training procedure involves performing 10 times with the same couple of C and γ values, but each time leaving out another one of the 10 parts as the validation set; the average prediction accuracy is calculated in the 10 runs. A grid of C and γ values are screened in order to determine which couple of values of C and γ gives the best accuracy. The best couple of hyper-parameters were found as $\{C, \gamma\} = \{16, 1.41\}$. The best classification rate, defined as the number of correctly classified features over the total number of image features, was 96.04%. The final step involves grouping the 10 parts to a single set and train it with the best couple of hyper-parameter values.

4.3 Accuracy of SVM prediction

The accuracy and the sensitivity of the SVM have been tested using 5 test images. Figure 5 illustrates one of these test image, where red and blue dots are the image features predicted as *vehicle* and *non-vehicle* class, respectively. The true positive classification rate (i.e. the total number of red dots on vehicles and blue dots on the background over the total number of the points in the image) was 92.16%. Over the 5 test images, there was a 0.58% false positive points (i.e. red dots on the background) and 7.26% false negative points (blue dots on the vehicles). Two vehicles located in the parking lot were missed (i.e. no red dots on the vehicles). Also, there is only one red dot point on three different vehicles. The sensitivity of the SVM, expressed as the ratio of red dots on vehicles over the total number of dots on vehicles (i.e. (*True Positives*) / (*True Positives* + *False Negatives*)), is equal to 54.8%. In the remainder of the paper, only true positives (detected red dots) will be considered.

Figure 5: Image features classified by the SVM, according to the feature vector elements. Red dots are classified as *vehicle* while blue dots classified as *non-vehicle*.

The collection of detected true positive features were clustered according to their relative metric distances into subsets with each subset associated to a vehicle. The UAPC algorithm was used to cluster the red points x with the negative square Euclidean metric distances $s(i, k) = -\|x_i - x_k\|^2$ as the similarity measure. Figure 6 shows the clusters of the true positive image features detected as *vehicle*. As showed by the red and blue rectangles, a few centers on the top of figure 6 are located over two neighboring vehicles.

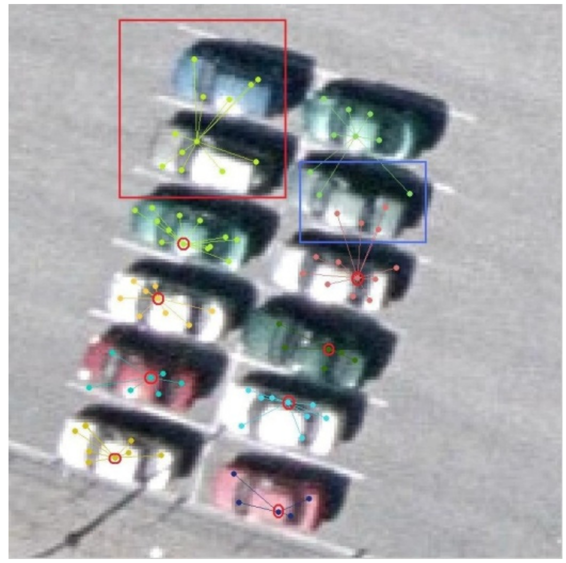


Figure 6: True positive image features clustered with multiple runs of UAPC algorithm without any constraints. A few centers are located over two neighboring vehicles (red and blue rectangles).

A way to cope with this problem is to add some constraints to the optimization problem. As a typical vehicle appears within a rectangle of 40×20 pixel size approximately at the spatial resolution of 11.2cm per pixel, the cluster spatial extension is limited to be no larger than 20 pixels to cluster only the red dots of a single vehicle. Accordingly, all values of $|s(i, k)| > D_{max}^2$ with $D_{max} = 20$ pixels are removed. This results in a new sparse similarity matrix such that the red dots with the distance values larger than D_{max} will no longer exchange messages and thus, will not be clustered.

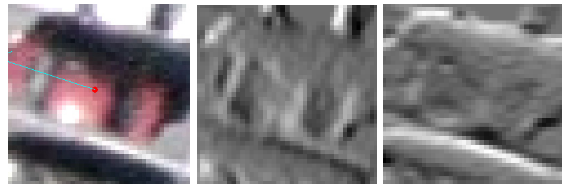
Moreover, a nonlinear similarity measure is introduced, where all similarities $|s(i, k)| < D_{max}^2$ are replaced by $-\frac{(s(i,k))^2}{(\frac{D_{max}}{2})^4}$. Therefore, red dots whose metric distance values range from 10 to 20 pixels will have their similarity (negative) values decrease as the power four of the distances, while those ranging from 1 to 20 pixels will be less affected. The nonlinear similarity measure will also help to avoid oscillation of the clustering caused by uncertain situations such as a point located between two other points with equal distances. In this particular case, the point located in the middle can be associated to either point alternatively in the iteration. Figure 7 illustrates the results generated using the new defined similarity matrix where the crossover clusters are eliminated by the UAPC algorithm with constraint on the cluster size.

Figure 7: Clustering results using only one run of the UAPC algorithm with constraints. The crossover clusters from fig.6 are eliminated.

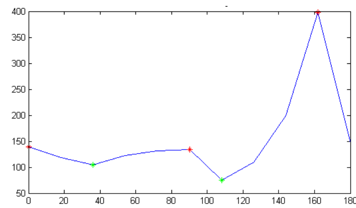
4.4 Multiple hits

The UAPC algorithm, used with a constraint on the spatial extension of the clusters being no larger than D_{max} together with some nonlinear similarity measures, generated clusters of image features that are located only on single vehicles (i.e. no crossover on multiple vehicles). However, as the vehicle size has been determined to be within an area of 20×40 pixels, it is possible to find more than one center point located on a single vehicle: this translate to a double or multiple hitting problem and increase the computational efficiency of the whole process.

In order to reduce further the number of centers generated, the orientation of the vehicles are used as an additional constraint. A patch of 40×40 pixels located at the center on each vehicle is extracted using only the gray scale image band. Gradient for each pixel of a patch are computed in order to build a histogram of 24 bins of the gradient orientation values. The maximum value in the orientation histogram is identified and then considered as the main direction of the selected vehicle. To avoid errors associated to low angular resolution of the image, each patch is rotated by $15°$ along 12 steps to build 12 histograms of the gradient orientations. The highest peaks in these histograms are selected as the vehicle orientation. Figure 8 illustrates the computed orientation of a single vehicle and associated horizontal and vertical gradient values. The computed orientation is $162°$ for this vehicle.



(a) Original (b) Vert. gradient (c) Hor. gradient



(d) Vehicle orientation $= 162°$

Figure 8: Vehicle orientation is computed using gradient values at each pixel of a 40×40 pixels patch. 8(a) Original subimage of the vehicle. Computed 8(b) vertical and 8(c) horizontal gradient values. 8(d) Computed vehicle orientation at $162°$.

A second rectangle window of 20×60 pixels is defined for each center point. If there is only one center in the window, than this center is associated to a single vehicle. If more than one center is located within a rectangle, they are fused together in order to represent a single vehicle. Using this approach, 43 vehicles from a total of 104 have been detected in figure 5. Blue squares in fig. 9 shows locations where vehicles are detected with only one cluster. For the cases where multiple clusters of located on a single vehicle, a weight is associated to each center representing the number of neighbors located on both its left and right sides. Then, a cluster is created for each center whose neighbors' weight is inferior, and then paired the remaining centers with the same weight values. At the end of the entire process, a total of 100 vehicles of the 104 ones are correctly detected with no false detection (Fig. 10).

5 Conclusion

A set of algorithms for the detection of vehicles in large scale aerial images was presented. Within a multiresolution linear Gaussian scale-space, developed algorithms extracted geometric and radiometric features from vehicles of an urban environment. Local structures describing the vehicles' image features of the scene were classified using support vector machines. Classified features were then clustered using an unsupervised affine propagation technique, where vehicle's subcomponents were unfolded. In order to cope with multiple hits cases where numerous features of vehicles were extracted, the orientation values of each feature were computed in order to reduce the number of features

Figure 9: Vehicles detected with only one cluster (blue rectangles).

Figure 10: Vehicles detected using the number of neighbors as additional constraints.

previously extracted. Experimental results using large scale aerial images demonstrated the efficient and the robustness of the proposed algorithms for the detection and counting of vehicles in an urban environment. Future work will involve the use of additionnal clustering techniques in order to improve further the extraction of vehicles in large scale aerial imagery of difficult and complex scenarios.

References

[1] H. Moon, R. Chellappa, and A. Rosenfeld, "Performance analysis of a simple vehicle detection algorithm," *Image and Vision Computing*, Vol. 20, No. 1, 1 January 2002, pp. 1–13.

[2] P. Burlina, V. Parameswaran, and R. Chellappa, "Sensitivity analysis and learning strategies for context-based vehicle detection algorithms," *DARPA IU Workshop*, Vol. 97, 1997, pp. 577–584.

[3] T. Zhao and R. Nevatia, "Car detection in low resolution aerial image," *Image and Vision Computing*, Vol. 21, No. 8, August 2003, pp. 693–703.

[4] S. Hinz, "Detection and counting of cars in aerial images," *IEEE International Conference on Image Processing (ICIP)*, Vol. 3, Barcelona, 2003, pp. 997–1000.

[5] C. Papageorgiou and T. Poggio, "A trainable system for object detection," *International Journal of Computer Vision*, Vol. 38, No. 1, 2000, pp. 15–33.

[6] H. Schneiderman and R. Kanade, "A statistical method for 3D object detection applied to face and cars," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2000.

[7] D. Lowe, "Object recognition from local scale-invariant features," *International Conference on Computer Vision*, Vol. 2, 1999, pp 1150–1157.

[8] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 27, 2005, pp 1615–1630.

[9] T. Lindeberg and B.H. Romeny, "Linear scale-space," *Geometry-driven diffusion*, Kluwer Academic Publishers, Dordrecht, Netherlands, 1994, pp. 1–77.

[10] J. Shawe-Taylor and N. Cristianini, "Support vector machines and other kernel-based learning methods," *Cambridge University Press*, Cambridge, 2000.

[11] B.J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, Vol. 315, 2007, pp. 972–976.

[12] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, Vol. 60, No. 2, 2004, pp. 91–110.

[13] V.N. Vapnik, "Statistical learning theory," *New York Wiley*, New York, 1998.

[14] C.J.C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery 2*, pp. 121–167, 1998.

[15] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, Vol. 20, Springer Netherlands Publisher, pp. 273–297, 1995.

[16] L. Tao and J.-J. Gu, "Active set iteration method for new L2 soft margin support vector machine," *Intelligent Computing in Signal Processing and Pattern Recognition*, Springer Berlin Publisher, 2006.