Volume 57, Number 4, 2016

ACTA TECHNICA NAPOCENSIS
Electronics and Telecommunications

EXPERIMENTING FEATURES SELECTION METHODS IN WEKA FOR COMPUTER VISION

Anca APĂTEAN
Technical University of Cluj-Napoca, Communications Department

Abstract: The paper presents the main possibilities to select features by the use of Search and Ranker methods in Weka, from monomodal or bimodal vectors comprising features extracted by 8 different algorithms. How to proceed to run and interpret the results within a number of 12 FS methods, by direct or n-folds CV application. By considering different threshold values for the feature rank, a number of 48 variants of FS methods resulted and these were applied to 6 different input vectors. All these were compared by the number of selected features and by the balanced accuracy as obtained with a KNN classifier.

Keywords: *features selection, crossvalidation, KNN classifier, Weka, Ranker, Search, balanced accuracy.*

I. INTRODUCTION

Nowadays intelligent systems are able, like humans interpret images, to acquire, process, analyse, and understand images by the use of computer vision techniques. Thus, a numerical or symbolic information is produced, e.g. in the form of decisions. To obtain the numerical representation of the data, which has to be recognized by the visible (VIS) - infrared (IR) obstacle detection and recognition (ODR) system [1], some features - comprised in feature families (FF) have been preferred, as already introduced in [1].

I. A. Features extraction and selection

The aimed features correspond to 8 different FFs and were obtained in the features extraction (FE) module. The first FF, denoted FF1, comprises only 2 geometrical features, i.e. the width and the height of the original bounding box enclosing the obstacle image (for both VIS and IR modalities); the other 7 FFs were separately extracted from each modality and they were: 7 statistical moments (FF2), 64 Haar wavelets (FF3), 32 Gabor wavelets (FF4), 8 Discrete Cosine Transform (DCT) coefficients (FF5), 16 grey level cooccurrence matrix coefficients (FF6), 14 run length encoding features (FF7) and 28 Laws features (FF8). Next, these FFs were combined to obtain the VIS and IR monomodal vectors, i.e. [FF1,FF2v,FF3v,…,FF8v] and [FF1,FF2i,FF3i, …,FF8i] and respectively the bimodal one [FF1,FF2v,FF3v, …,FF8v, FF2i,FF3i, …,FF8i], with v and i specifying the image type from which those features were extracted. Computed in this way, the monomodal vectors comprises 171 features, while the bimodal one has 340 features. In order to decrease the system processing time, so to accomplish a real time request, one question arises: *How to decide which are the most representative features to describe the data in our classification system*? One possible answer is: by a *features selection (FS)* operation. Still, multiple FS methods exists, with multiple setup possibilities (and in our case also applied to multiple possible feature vectors (FVs)), so *How to apply these FS methods?*

The first tested FS methods, as presented in [1], were the *Correlation-based Feature Subset (CFS)* and the *Consistency Subset Evaluation* (*CSE*). These features evaluation methods were combined with a *Best First search.*

After obtaining the new vectors, containing only the retained features (organized as sFVs) two possible problems appeared: *These new sFVs will maintain the high accuracy rates*? or, if these will degrade, *How can we compensate this?* After performing the experiments presented in [1], the solution to compensate that accuracy loss was by exploiting *the SVM model selection* (i.e. by the kernel type and the hyper-parameters selection). A bi-optimization criterion was used and the designing aspects referred to how the features selection (FS) and kernels selection (KS) were accomplished to speed up the system. The solution proposed in [1] aimed the final stages of the system design and imply the use of an SVM, as also presented in [2].

In this paper, in order to evaluate more the FVs and to compare them in a direct and rapid manner, *a simpler classifier*, i.e. *a KNN (K-Nearest Neighbor)* one was preferred. This does not have hyperparameters to be optimized, like is the case of the SVM.

Generally, two operations are important for a classification problem:

(1) finding the *proper FV* to characterize the data and

(2) finding the *best classifier* to process the respective FV. Unlike in [1], when the subject was more around the second task, this paper concentrate on the first one.

Generally, a classification system uses a training set of data with examples or observations whose category membership is known in order to learn how to discriminate new observations. A *bimodal* classification system considers two different types of data (VIS and IR images in our case) in taking the final decision. The way it will be able to process and interpret these images by using *modality independent features vectors* (monomodal ones extracted from the corresponding modality) or by using *bimodal features vectors* (extracted after the modalities were combined) is referred in this paper. Different types of classification systems exist, some comprising sets of data which cannot be interpreted in the same way (so the system is able only to manage more monomodal subsets or FVs of data), while others allow it (by using multimodal vectors).

Manuscript received November 3, 2016; revised December 13, 2016

I. B. Using Weka to perform the FS task

Weka (stands for the Waikato Environment for Knowledge Analysis) includes a collection of ML algorithms for data mining tasks. It was developed in Java and belongs to University of Waikato in New Zealand [3]. The algorithms implemented within Weka can be applied directly to a dataset or called from an own Java code (natively) but also Matlab or Python code after installing specific libraries.

An extensive manual for using Weka can be followed in [4], but a more comprehensive one (including theory behind algorithms implementation in Weka) is in [5].

The *Explorer GUI* is generally used to load the data in Weka, apply some pre-processing on that data, select features, use one of the available ML algorithms, and finally visualize the built models or the results. In the experiments presented here, only the Explorer part was framed. There are also the Experimenter and the Knowledge Flow options, as presented in version 3-7 from the Weka official site [3], where both a platform-specific installer and an executable Java jar file can be found.

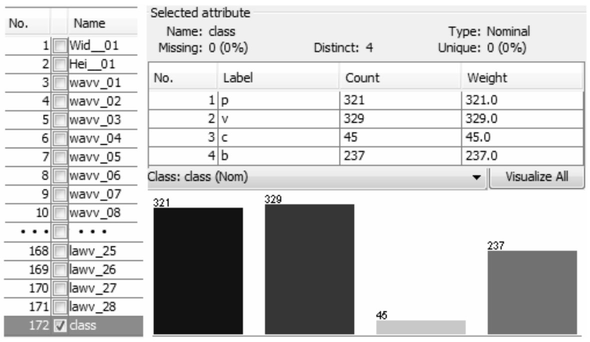
Weka was used not only to perform the FS task, but also to apply the KNN to the dataset and analyse its output. Before applying any classification algorithm to the data, this must be converted to the ARFF form with type information about each feature or attribute as called in Weka - Figure 1. In the performed experiments, a MATLAB Version for LIBSVM [6] and an own developed code were used.

II. MODALITY INDEPENDENT VS BIMODAL FEATURE VECTORS

II.A. Applying the FS method on different vectors

The FV construction generally comprises a FE and a FS tasks. In this section, two types of FVs were considered, which were called monomodal and bimodal and which depend on the position the modules performing these tasks are positioned in the scheme of a complete computer vision based system.

The monomodal FV is the one obtained on the VIS or on the IR modality individually; therefore, it could be considered as a type of FV dedicated to a *monomodal* system (i.e. a system processing a single type of data). Unlike these systems, are the *bimodal* ones, which are capable of processing two types of information (in our case VIS and IR). The benefits are for both sides. The *monomodal ones* are not so constrained considering the processing time, because they can manage the data in a parallel way. Still, the performances are generally higher in the second case, i.e. that of *bimodal systems*.



No.		Name
1	☐	Wid__01
2	☐	Hei__01
3	☐	wavv_01
4	☐	wavv_02
5	☐	wavv_03
6	☐	wavv_04
7	☐	wavv_05
8	☐	wavv_06
9	☐	wavv_07
10	☐	wavv_08
• • •	☐	• • •
168	☐	lawv_25
169	☐	lawv_26
170	☐	lawv_27
171	☐	lawv_28
172	☑	class

Selected attribute
Name: class Type: Nominal
Missing: 0 (0%) Distinct: 4 Unique: 0 (0%)

No.	Label	Count	Weight
1	p	321	321.0
2	v	329	329.0
3	c	45	45.0
4	b	237	237.0

Class: class (Nom)

Figure 1. An arff file with a monomodal vector in Weka

The FS task could be accomplished:

1) immediately after the *individual FVs* have been obtained from the FE step, and monomodal FVs result; to these, the FS operation can be applied: they will be referred with an "i" at the power index, showing that the FS operation has been applied on *VIS and IR vectors individually* or

2) after the extracted features have been combined in a bimodal FV and in this case the resulted FV is a *concatenated FV*: it will be denoted with a "c" at the power index, because the FS operation has been applied on the *concatenated VISIR vector*, as presented in Figure 2. Here, the features fusion module does not refer to a fusion of more features or families of features belonging to the same modality, but to an intra-modality fusion of features.

Figure 2 could be split in two different drawings: one for the *individual case* - where the fusion is applied to modality independent FVs (the right-bottom side of Figure 2) and another one for the *concatenated case* - where the fusion is first performed to the initial monomodal vectors and only after is applied the FS operation, thus on a bimodal FV (the left-bottom side of Figure 2).

These approaches have been introduced in [2], where also their utilization in the frame of the proposed ODR system is mentioned. In this paper, only the way these individual or concatenated FVs were obtained is presented.

The experiments concerning the FS stage aimed to simply use these sFVs to represent an obstacle image, feeding later a classifier. Thus, the purpose of these investigations was to evaluate their performance in a similar way as realized when creating the initial FV, in the FE module. A simple classifier was thus required.

II. B. Different notations for the selected FVs

The two types of sFVs are further needed in the last module of the ODR system, i.e. to perform the fusion task, their purchase being very much influenced by the mode the FS scheme has been applied.

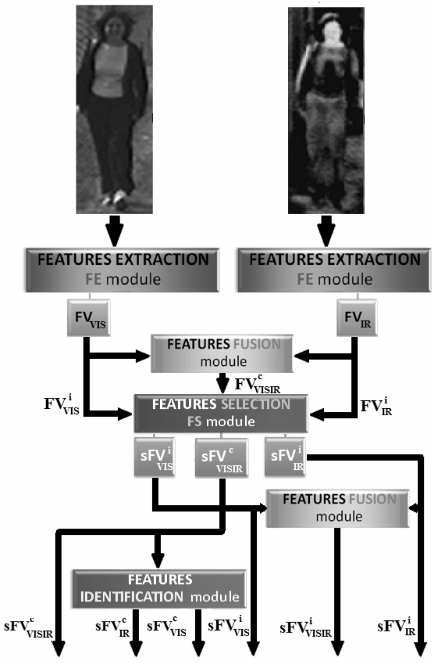


Figure 2. FS from modality independent and bimodal FVs

The results obtained in the two previously presented situations are generally different, so it is important to mention the way the FS method was applied in a bimodal system. It was denoted *sVISIR*^{c} the bimodal vector which contains the features from both domains, VIS and IR, after the FS operation has been performed on the *concatenated (bimodal) vector VISIR*. By the separation of the features in monomodal vectors, the selected features-vectors, denoted *sVIS*^{c} and *sIR*^{c} were later obtained. In the case the selection scheme was applied on the *individual monomodal vectors*, the obtained FVs were *sVIS*^{i} and *sIR*^{i}, and they were later combined in the bimodal vector *sVISIR*^{i}.

The notation $\text{VIS}^c{}_{17}$ for example, denotes that the respective feature vector contains 17 selected features corresponding to the VIS domain, these being provided by the concatenated approach.

In the case of the individual selection, if the same common feature, e.g. *width*, has been selected on both modalities VIS and IR, it was inserted in the final bimodal sFV VISIR only once. Still, a feature could have been inserted two times in the sFV to double its importance when using specific classifiers, but here such a situation was not considered.

III. THE CLASSIFICATION PROBLEM

In the FS stage, the importance of the features extracted in the FE module and later combined in a single FV is estimated. Only the most relevant features were chosen to further represent the information, the resulted vector being denoted sFV. In order to perform the FS task, Weka was framed [3] and the FS algorithms were applied by controlling Weka from Matlab.

The database was randomly divided into a training set (80%) and a testing set (20%), the instances being well balanced between these two sets. A specific FV was extracted for each of the 321 pedestrians, 329 vehicles, 45 cyclists and 237 background objects (the training set, as shown in Figure 1). For more details and information, please consult [1], [2]. The database was originally designed for pedestrian detection from four stereo correlated VIS–IR images [7], [8] and manually annotated.

To evaluate the classification accuracy of the system, an arithmetic average of class accuracies, which is called a *balanced accuracy (bAcc)* was used; this is a particularly useful evaluation measure for unbalanced datasets, being defined by the equation (1):

$$bAcc = \frac{1}{K}\sum_{i}^{K} F(y_i \mid y_i) = \frac{1}{K}\sum_{i}^{K} TPR(i) \qquad (1)$$

where K is the number of objects classes and TPR is the true positive rate computed for each class.

Because the data used in our experiments are not balanced in classes (e.g. there are 36% vehicles and only 5% cyclists), in the performances computation, the *bAcc* was used instead of the weighted *Acc* (as computed in Weka). An example for the classification problem with 4 classes of objects is presented next, in order to motivate this choice of *bAcc* instead of *Acc*. Suppose a classifier provides the confusion matrix from Table 1. The summary of the results from the training data ends with such a confusion matrix, showing how many instances of a class have been assigned to each possible class. If all instances were correctly classified, only the diagonal elements of the matrix are non-zero. The recall or *TPR* will be computed for each of the four classes as: *TPR = TP/ (TP+FN)* with TP the true positive value and the FN the false negative one. For example, the following values were reached for the pedestrian class *TPR*(*P*) = 283/321 = 0.882, for the vehicle class *TPR*(*V*) = 282/329 = 0.857, and so on. The value *Acc* was computed like in Weka and it is:

Acc = (283 + 282 + 0 + 186)/932 = 0.806.

On the other side, if the *bAcc* was computed to respect the equation (1), resulted:

bAcc = (0.882+0.857+0+0.785)/4 = 0.631.

Table 1: An example of the confusion matrix (CM) obtained for a classification problem with 4 classes of objects

CM	**P**	**V**	**C**	**B**
P	*283*	*17*	*0*	*21*
V	*20*	*282*	*0*	*27*
C	*20*	*25*	*0*	*0*
B	*27*	*24*	*0*	*186*

This confusion matrix highlighted that there is a problem at the classification: the classifier was not able to correctly learn the C class, so cyclists were not recognized at all. In this way, *bAcc* was preferred instead of *Acc*, the value 0.631 not 0.806 being used (to measure the system accuracy); bAcc was more appropriate to show that something has gone wrong at the classification. An obstacle recognition system which does not recognize any cyclist from the road has certainly a big issue and a solution must be found.

The features (extracted by different algorithms and from different modalities) were quite different as concerns their representing scales; in order to be properly combined in a multimodal vector, they have been normalised in the same domain, prior to the FS task.

IV. SELECTING FEATURES IN WEKA

IV.A. FS methods and the modalities they apply to

By adding irrelevant attributes to a dataset often distract or confuses ML based systems. Instance-based ML algorithms (like KNN) are very susceptible to irrelevant attributes because they always work in local neighbourhoods (these take just a few training instances into account for each decision). The number of training instances needed to produce a predetermined level of performance for these ML algorithms increases exponentially with the number of irrelevant attributes, as stated in [5].

Reducing the dimensionality of the data by deleting unsuitable attributes, generally improves the performance of ML algorithms. This operation of dimensionality reduction yields a more compact, but also a more easily interpretable representation of the target concept.

In addition, when considering all features (even the unsuitable ones), besides the poor accuracy rate, even the classification time could be very much degraded, i.e. increased, compared to a situation in which a small set, of only relevant features is used. Once a good small set of features has been chosen, even the most basic classifiers (e.g. KNN) can achieve high-level performances.

For the FS operations, there are multiple variants, concentrated on two fundamental directions: filters and wrappers. These differ mostly by their evaluation method. In our experiments, only filters were used, as they are generally faster. For any filter method, the type of the *attribute evaluator* together with the method to be applied, to *individual features*, as for the *Ranker* methods or to *subset of features*, as for the *Search* methods, should be mentioned.

Cross-validation (CV) could be also applied for FS (not only for model selection), since CV provides an estimate of the generalization ability of models.

The use of the CV process also in the FS stage was considered, even the time needed for the algorithm to perform the FS was increased, as introduced in [1]. The processing of FS methods by CV, with n=100 was performed in a loop and takes much more time than another method not using it. This is not critical for our system, it is not even affecting it, because no real-time operation is required in this stage; the FS operation is performed off-line, when the system is not running on the road (just prepares for the online functioning).

IV.B. Applying FS methods in Weka

In Weka, the FS methods could be applied:

(1) directly, *on the full training set (TS) of data:* the FS method is applied only once on all the data from the TS, or

(2) *by a cross-validation technique:* the FS method is applied on each individual fold of data, the union of the folds composing the training set; there is a number equal to the number of folds (100 in our case) for the application of the respective FS method.

The first situation, denoted *FullTS,* is illustrated in Figure 3a), while the second one, denoted *100f-CV* is illustrated in Figure 3b). These selection procedures may lead to different subsets and, depending on the application and the classification problem objectives, one approach may be preferred over another.

IV.C. Ranker and Search FS methods

As suggested in [9], two types of *search approaches* (*Ranker* and *Search*) and two types of *attributes evaluators* (*single-attribute* and *subset-attribute)* were used (their summary is shown in Tables 2 and 3).

Attribute selection can be performed by searching the space of attribute subsets, evaluate each one and finally choose the best combination (based on an evaluation measure). This is generally achieved by combining *one attribute subset evaluator* with one search method, as presented for ***Search methods***.

Another potentially approach is to evaluate the attributes individually, sort them, and later discard some of them (e.g. those falling below a chosen threshold).

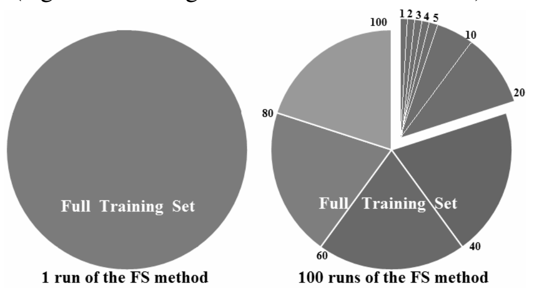


Figure 3. The application of the FS method, on the FullTS a) once and b) one time on each of the 100 folds

This method of selecting features does not perform a selection by its own. In Weka, it could be achieved by selecting one *single-attribute evaluator* with a ranking method, as presented in the case of the ***Ranker methods***.

Single-attribute evaluators used with the Ranker search method generate a ranked list of features. The main idea is to order the features regarding to a weight value (or the feature importance) which could be determined by some characteristic of the data. Many criteria could be used in exploiting the data characteristic, like [5] states, but here the most popular ones were chosen: *Chi Squared, Information Gain, Information Gain Ratio, ReliefF, Significance* and *Symmetrical Uncertainty* as shown in Table 2.

Table 2. Ranker-based Features Selection (FS) methods

Evaluators	**Single-attribute Evaluators**					
	Chi Squared	***Information Gain***	***Ratio of the Info. Gain***	***Relief F***	***Significance***	***Symmetrical Uncertainty***
Ranker	FS1TS	FS2TS	FS3TS	FS4TS	FS5TS	FS6TS
Selection Mode	***Full TS***					

By selecting a single-attribute evaluator combined with Ranker, a potentially faster but less accurate approach is generally reached. The *Ranker* method evaluates the individual attributes and sorts them, providing thus an ordered list. In this way, naturally results two possibilities of discarding some attributes: 1) a cut-off threshold (below which attributes are discarded) or 2) the number of attributes to be retained.

On the other side, by combining one *attribute subset evaluator* (like CFS or CSE) with one *Search method* (for example *Best First, Linear Forward* and *Genetic Search*) as shown in Table 3, FS is normally done by searching the space of attribute subsets and choosing the best one based on an evaluation measure or stopping criteria [5]. *Search methods* get through the feature space to find *a good subset*, its quality being measured by an attribute subset evaluator. *Subset evaluators* take a subset and return a numeric value (or measurement) that will be further used to guide the search. The approaches combining a feature *search method* and *a feature-subset evaluator* measures the quality of a descriptor by considering also the other attributes. They generate some candidate solutions by the subset searching method and evaluate the performance with the associated evaluator (a weight for the generated subset is assigned for a comparative analysis between all the generated subsets).

Table 3. Search-based Features Selection (FS) methods

Evaluators		**Subset-attribute Evaluators**			
		Correlation (CFS)		***Consistency (CSE)***	
Selection Mode		***Full TS***	***100f-CV***	***Full TS***	***100f-CV***
Search	**Best First**	FS7TS	FS7CV	FS10TS	FS10CV
	Linear Forward	FS8TS	FS8CV	FS11TS	FS11CV
	Genetic Search	FS9TS	FS9CV	FS12TS	FS12CV

Generally, one of the two following methods are used in the literature as an attribute subset evaluator:

(1) the *Correlation-based Feature Subset* (CFS) which evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them [5], [10];

(2) the *Consistency Subset Evaluation* (or simply CSE) which evaluates the worth of a subset by the level of consistency in the class values when the training instances are projected onto that subset [5].

In Table 2 and Table 3 there is also mentioned *the attribute selection mode*, so to which FS methods corresponds and in what manner this was applied.

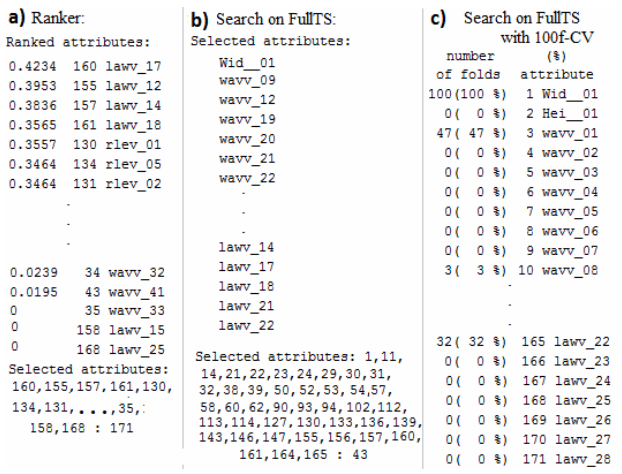
To summarize, all the tested FS methods were grouped in two sets: the first set, with FS1…FS6 and the second set containing the other 6 methods, denoted FS7…FS12. The last set is further divided in two groups, depending on the procedure the FS methods have been applied on the data. In both cases of the FS method application (with the data illustrated in Figure 4) the same dataset, i.e. the TS, was used when applying the FS.

IV.D. Interpreting the results of the FS methods

The information schematically presented in Figure 4 was obtained by running:
- one Ranker FS method (a)) and
- one Search FS method (with both selection modes - b) *once on the entire FullTS* and c) *on the FullTS by 100f-CV*).

With Ranker: The FS method for the first set, as presented in Table 2, i.e. for those based on *Ranker* (the cases FS1 … FS6) and denoted FullTS, suggest that the FS method was applied using the full training set, only once on all the data from the TS. In this way, there is a single run of the respective FS method, as suggested by Figure 3a). By the nature of the Ranker algorithm, a list of ranked features was provided as the result, as shown in Figure 4a).

The first attribute from this list (shown in Figure 4a)), has the highest importance, the second one from the list has a lower importance and so forth, the importance being decreased as proceed further to the end of the list; some of the features from the very bottom were even indexed with a zero rank value.


a) Ranker:

```
Ranked attributes:
 0.4234  160 lawv_17
 0.3953  155 lawv_12
 0.3836  157 lawv_14
 0.3565  161 lawv_18
 0.3557  130 rlev_01
 0.3464  134 rlev_05
 0.3464  131 rlev_02
           .
           .
           .
 0.0239    34 wavv_32
 0.0195    43 wavv_41
 0         35 wavv_33
 0        158 lawv_15
 0        168 lawv_25
Selected attributes:
 160,155,157,161,130,
 134,131,...,35,
   158,168 : 171
```

b) Search on FullTS:

```
Selected attributes:
    Wid__01
    wavv_09
    wavv_12
    wavv_19
    wavv_20
    wavv_21
    wavv_22
       .
       .
       .
    lawv_14
    lawv_17
    lawv_18
    lawv_21
    lawv_22
 Selected attributes: 1,11,
  14,21,22,23,24,29,30,31,
  32,38,39,50,52,53,54,57,
  58,60,62,90,93,94,102,112,
  113,114,127,130,133,136,139,
  143,146,147,155,156,157,160,
      161,164,165 : 43
```

c) Search on FullTS with 100f-CV

```
  number       (%)
 of folds    attribute
100(100 %)    1 Wid__01
  0(  0 %)    2 Hei__01
 47( 47 %)    3 wavv_01
  0(  0 %)    4 wavv_02
  0(  0 %)    5 wavv_03
  0(  0 %)    6 wavv_04
  0(  0 %)    7 wavv_05
  0(  0 %)    8 wavv_06
  0(  0 %)    9 wavv_07
  3(  3 %)   10 wavv_08
           .
           .
           .
 32( 32 %)  165 lawv_22
  0(  0 %)  166 lawv_23
  0(  0 %)  167 lawv_24
  0(  0 %)  168 lawv_25
  0(  0 %)  169 lawv_26
  0(  0 %)  170 lawv_27
  0(  0 %)  171 lawv_28
```

Figure 4. Results of the application of a) Ranker; b) Search on FullTS; c) Search on FullTS with 100f-CV.

In our case, the highest rank value was 0.4234, this being in top of the list; it was assigned to the feature numbered 160th in the initial FV, which is the seventeen-th from the law FF from the VIS domain.

This rank value continues to drop, for each element down the list, the last three ones from the very bottom being ranked with 0 (not illustrated in Figure 4a)), so they are considered to be not relevant.

It is important to underline that all the features from the initial FV are comprised in this list and they appear only once within the list.

With Search – once on the entire FullTS: Unlike this, for the second set of the FS methods, as presented in Table 3, i.e. for those based on *Search* (methods FS7 … FS12) if the FS method would be applied directly on the Full TS (and only once), the result would be a subensemble of features, thus a subset of the initial one.

This subset (containing thus only the selected features) is provided alphabetically ordered, in function of features name, no level of their importance being mentioned.

As it could be noticed in the example presented in Figure 4b), the features are ordered by their family, and in each family, they are listed by their ordinal number. For example, from the first FF, i.e. the *geometrics* one, only width was selected, which is the first feature from the initial FV. Next, from the second FF, i.e. the *wavelet* one, multiple features were retained: 09th, 12th, 19th, etc, which are the 11th, the 14th, and respectively the 21st from the initial FV comprising all the 171 VIS features. Therefore, no information about the importance of the features in the subset will be given by applying in this way the Search method. In addition, the number of selected features could be very small, so no intermediate approach will be possible.

With Search – on the FullTS by 100f-CV: By the application of the CV method on 100 folds, such an index of relevance has been obtained also in the case of methods FS7 … FS12, as in the case of methods FS1 … FS6 (those provided directly by the FS method); this index of relevance was available by the application of the FS method by a CV procedure, and thus, the features were more easily evaluated individually. In this way, the FS process may be repeated for many sub-samples of the training data as illustrated in Figure 3b) and the union of the subsets of selected features may be taken as the final subset.

An index of relevance of individual features has been created considering how frequently they appeared in the selected subsets. By including the FS process in the CV loop, the number of selected features has been determined as the number of elements of the reunion of all sets selected in each of the 100 folders of the CV loop.

To produce the final FV, it is taken into account the same data from the Full TS, but as comprised in 100 folds, as shown in Figure 4c). By a proper arranging of these selected features, starting from those selected in all 100 folds, then going on with the ones selected in 99 folds, and so on, until the ones not selected in either fold, a similar list of ranked features has been obtained, as shown in Figure 5. This ranked list is somehow similar with the one from Figure 4a), but this time the rank value is indirectly computed, thus provided by the number of folds in which that feature appeared as relevant, so selected.

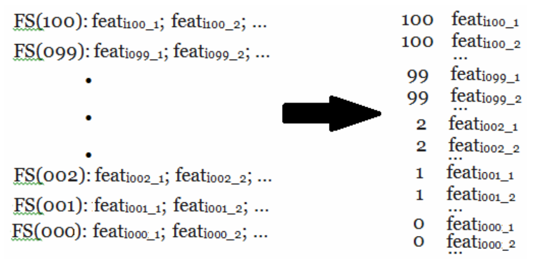

FS(100): $feat_{i100\_1}$; $feat_{i100\_2}$; ...
FS(099): $feat_{i099\_1}$; $feat_{i099\_2}$; ...
FS(002): $feat_{i002\_1}$; $feat_{i002\_2}$; ...
FS(001): $feat_{i001\_1}$; $feat_{i001\_2}$; ...
FS(000): $feat_{i000\_1}$; $feat_{i000\_2}$; ...
100 $feat_{i100\_1}$
100 $feat_{i100\_2}$
99 $feat_{i099\_1}$
99 $feat_{i099\_2}$
2 $feat_{i002\_1}$
2 $feat_{i002\_2}$
1 $feat_{i001\_1}$
1 $feat_{i001\_2}$
0 $feat_{i000\_1}$
0 $feat_{i000\_2}$

Figure 5. Rank values obtained by the application of the Search FS methods (FS7…FS12) by 100f-CV.

If two FS methods provide two different FVs, but with the same accuracy rate, the one having a smaller number of features in the FV should be chosen as the best one. A smaller number of features could lead to the elimination of some entire families of features (especially if the reduction is a severe one) and in this case, the time to compute the new FV should be smaller.

Another possibility to reduce the extraction time is that when discarding the features, some of them to be separable ones. In the case they are not separable, the best scenario regarding this aspect will be that: if a majority of features from a family is retained or discarded (before ending the FS process), then all the features from that family to be retained, respectively discarded by the moment when that FS operation will stop; this stands for applying a kind of an heuristic rule implemented in the FS loop: to retain all selected features or none from that family, based on a majority threshold criteria.

To conclude, the methods previously presented in Tables 2 and 3 were investigated: some of them are capable to provide an index of relevance directly by the classic FS approach (FS1… FS6), while some of them can give such an index only indirectly, by the use of the CV procedure (FS7… FS12). If in the case of the methods FS1 … FS6 we have to use thresholds to select a subset of features from the entire set, we will also have at our disposal this possibility for the methods FS7 … FS12, both methods providing rank values. These thresholds will be employed to select more or less features from the ones already rendered by the FS method, as one may need.

Next, the evaluation of all the methods and their variants schematically given in Tables 2 and 3 is presented.

V. EXPERIMENTS

V.A. Aspects concerning the experiments setup

A critical aspect of a FS method is to properly assess the quality of the selected features. The obtained FVs after the FS process, thus sFVs will be evaluated and compared with the initial ones. We recall that the initial monomodal FVs have a dimension of 171 features (VIS_{171} and IR_{171}) and the initial bimodal FV has 340 features ($VISIR_{340}$). The comparison with these values will be presented next.

From the beginning, the use of an SVM was aimed in the final implementation of the ODR system (as already presented in [1]), as its capabilities are one of the most appreciated in the ML community. Still, when was about to evaluate the FVs, not the classifier, a KNN was chosen: the exploited classifier is a KNN one with K = 1, for the simplicity in its usage; besides, it does not require a parameter optimization process (like the SVM does). In addition, the main purpose in this step was to optimize the FV, not the classifier hyper-parameters. The performance criterion is the balanced accuracy (bAcc) obtained on the training dataset, when a 10 folds cross-validation (10f-CV) procedure was performed.

In a 10-fold cross-validation process, the original sample of data is partitioned into 10 sub-samples. From these 10 sub-samples, a single sub-sample is retained (the validation or testing set), and the remaining 9 sub-samples are used to train the classifier (the training set). The cross-validation process is then repeated 10 times and a combination of the 10 results (generally the average) is performed in order to obtain the accuracy value.

Next, the performances obtained by the KNN when using 10f-CV for the FVs obtained after the application of different FS methods will be compared. This means that first, the FS method has been applied on the data from the training set and some new sFVs were obtained. Next, with the information encoded by these sFVs, a KNN classifier has been used by a 10f-CV procedure on the same training dataset for tests. Only in a later experiment, it was also used in the learning-testing stage, so learn all the instances from the learning set and test with a new, unseen testing set.

In this paper, is not performed a deep insight into the Weka functionalities, but only aspects regarding the FS operations are envisioned. Next, the results of the developed experiments are presented.

V.B. Interpreting the obtained results

The evaluation of different FS methods available in Weka has been achieved by combining multiple features, modalities, application and types of FS methods, setup parameters, etc. The main purpose was to test multiple types and setup variants to accomplish the FS task (combining more search and features evaluation strategies) and finally to propose a way to retain only the best one. The solutions proposed in this paper implies to find a common setup for multiple FS methods in order to intervene at an intermediate level in the FS task.

Because each category of 6 FS methods (Ranker and Search) provide a ranked list of features, it is expected that all the features to present a rank value, a strict positive number. By the normalization of this rank value in the domain [0, 100], the selection thresholds could be chosen equally distributed in the specified domain. Thus, in order to select only the most important attributes from the ranked list provided by a specific FS method, 3 threshold values have been proposed: the corresponding FS methods have been denoted FSn_Thr, where $n \in [1;6]$ and $Thr \in \{25,50,75\}$.

There are multiple possibilities to retain features based on such a threshold parameter, but in this paper the next interpretation was used: it may denote the value of the rank associated to each feature (as made available also for the Search methods not only for the Ranker ones). Each category of the 6 FS methods mention three values for the parameter threshold. In fact, each of the 3 variants, having associated a threshold value, starts from the ranked list provided by the application of the respective FS method on the full training set and retains features as specified by the value of the threshold.

Still, a fourth variant (in each of the category) where the FS would have been applied on the FullTS and the threshold would have been 0 (interpreted as "select all features having a rank value higher than 0") could be considered. This situation would have been exactly the same with the one selecting all features and it could have been considered as a reference for the other 3 variants of each group of 6 ones. Figures 6, 7, 8 and 9 present the obtained results.

The threshold values refer to a rank value (denoted rankThr) which has to be overcome in order that a feature to be retained in the selected set. For example, the method using the threshold at the value of 75, will select a smaller number of features than the method using a threshold of 50, and this latter will select a number even smaller than the one employing a threshold of 25. In this last case, all the features having a rank value higher than 25 will be retained. If a rankThr of value 0 would have been chosen, all the features from the initial FV were retained (like no FS was applied).

Figures 6 and 8 shows the dimension of each vector (the sFV one) after the application of the respective FS algorithm, i.e. in the Ranker or the Search case respectively. Both figures illustrate the size of the new FV which is better if it is smaller. Figures 7 and 9 shows the balanced accuracy obtained by the KNN classifier (K=1) when using the 10f-CV procedure applied on the training set. The data from the training set has been characterized only by the features retained after the application of the respective FS operation. Both figures illustrate the accuracy of the sFV, which is better if it is higher. A method which neither provides good accuracies, nor the number of retained features is small, should not be selected. In Figures 6 and 8, a too small value for the number of retained features in the sFV (e.g. for those with a threshold of 75 in the Ranker case or the ones FS10, FS11, FS12 in the Search case) may lead to poor accuracy rates (the corresponding areas in Figures 7 and 9). In the Ranker case, the poorer result is with VISc vector which in the majority of the cases with a threshold of 75 reached an accuracy of 0. This is explained by considering the number of the selected features: 0,0,1,0,5,0 for FS1…FS6. By considering a threshold of 50, the same vector retains 5, 2, 25, 12, 93, 6 features and with a threshold of 25 it retains 64, 60, 104, 88, 165, 81 features. A similar analysis can be performed also for the IRc or VISIRc vectors, but also on the individual selection cases, and not only for Ranker (Figures 6 and 7), but also for Search (Figures 8 and 9).

Only few sFVs from the individual selection succeeded to provide an accuracy higher than 97%*Accmax (where Accmax is the accuracy obtained by the corresponding vectors comprising all features). The following sFVs stand out: FS1_25, FS2_25, FS3_25, FS4_25, FS5_25, FS5_50, FS6_25. Generally, only the sFVs with a rank value higher than 25 succeeded to accomplish this *first criteria*. Even more, there are sFVs which accomplished the criteria even with 110 features (FS1_25, with 110, 77, 186 features on VIS, IR and VISIR respectively provided 85%, 86.3%, 92% accuracy) or 118 features (FS5_50 with 118, 86, 203 features provided 85.5%,87.4%, 93.4% accuracy).

As concerns the results from the concatenated selection, only next sFVs stand out: FS3_25, FS4_25, FS5_25, FS5_50. The FS5_25 vector used the highest number of features on every modality and provided not much improved results (84, 88.7, 93.3 accuracy compared with 83.7, 88.0, 94.4 the reference obtained for the vectors AllFeatures; the number of features was very high: 165, 169, 332, selecting almost all features (96% from VIS, 99% from IR and 98% from VISIR). Thus, it was not considered in the final stage of the system.

Compared to the ones corresponding to the Ranker methods, the Search ones are able to accomplish the criteria based on which a FS method is further retained with a much smaller number of features than their Ranker counterparts. The size of the sFVs is reduced with at least 25% in these cases and the sFVs are still able to provide accuracies which overcomed the ones obtained with the initial vectors VIS171 and IR171. It has to be noticed the dramatic reduction of the FV size in the case of the method FS7CV, which with only 38% respective 36% features on VIS, respectively on IR domains (i.e. 65 and 62 features), provides accuracies higher with at least 1% than those rendered by the vectors AllFeatures.

By the use of the KNN, the obtained accuracy values were smaller than those obtained with the initial FVs, but compensation solutions were envisioned, as presented in [1], [11], [12], [13], in the second stage of the classification, i.e. when searching for the best classifier or fusion method to improve the FVs outputted from the first stage. Instead of choosing a single FS method from the beginning, in this paper a selection of the FS method adapted to the initial FV was proposed. A number of 12 FS methods were evaluated, but by considering different variants to retain the features in the final sFV (by the use of the threshold), a number of 48 FS variants were reached. By considering their application on individual or on concatenated FVs a number of 6 possible FVs were evaluated for each of the 48 FS variants.

VI. CONCLUDING REMARKS

To summarize, the initial FVs were constructed by different FE algorithms, which we called feature families. Next, new FVs were obtained by the application of the FS method. In this paper, three search methods (i.e. Best First, Linear Forward and Genetic Search) were used in combination with an attribute subset evaluator (CFS and CSE) and the number of selected features was determined by the combination of these two procedures. To exploit the data characteristic, the most popular methods, i.e. Chi Squared, Information Gain, Information Gain Ratio, ReliefF, Significance and *Symmetrical Uncertainty* were chosen in combination with the Ranker method (which evaluates features individually).

Because it is not known beforehand which method of FS will provide best results, instead of selecting a single FS method from the very beginning, in this paper was proposed to apply multiple FS methods, to evaluate their results and only after, to choose the proper one for a specific problem. Thus, also a selection of the FS methods was accomplished. We preferred to use the FS methods already available in Weka, because Weka implements a multitude of ML algorithms and more important is that these algorithms are consistently updated and improved.

As further improvements, the use of the regression analysis to study possible relationships between features is envisioned, so to understand which among the independent variables are related to a dependent variable, and to explore the forms of these relationships. A possible starting point is the comparative study of different types of regression (linear, polynomial, spline and B-spline) from [14].

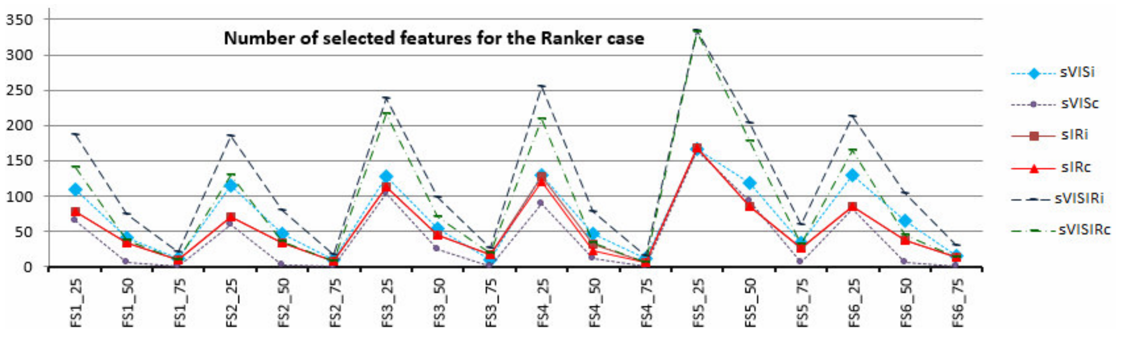


Figure 6. Number of selected features for the Ranker type of FS methods (FS1…FS6)

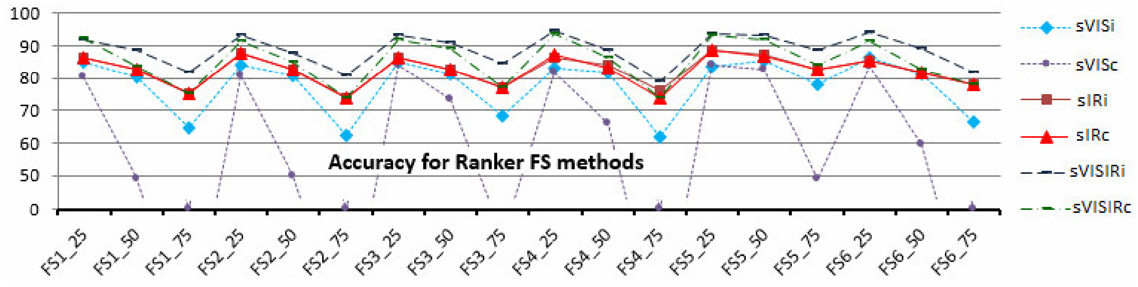


Figure 7. KNN accuracy for the Ranker type of FS methods (FS1…FS6)

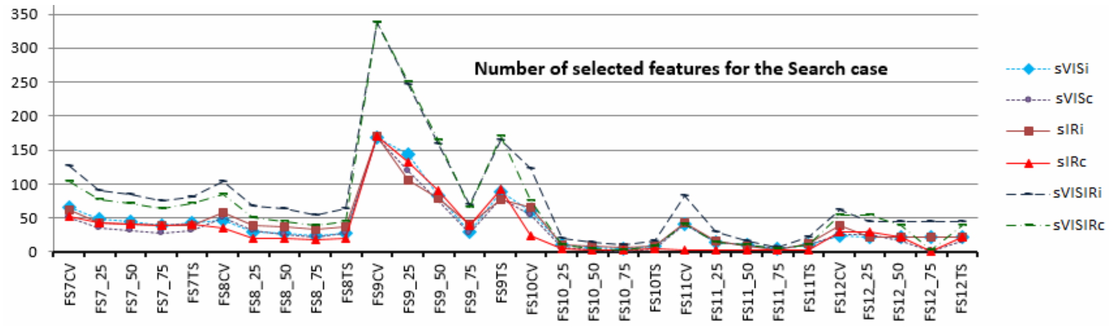


Figure 8. Number of selected features for the Search type of FS methods (FS7…FS12)

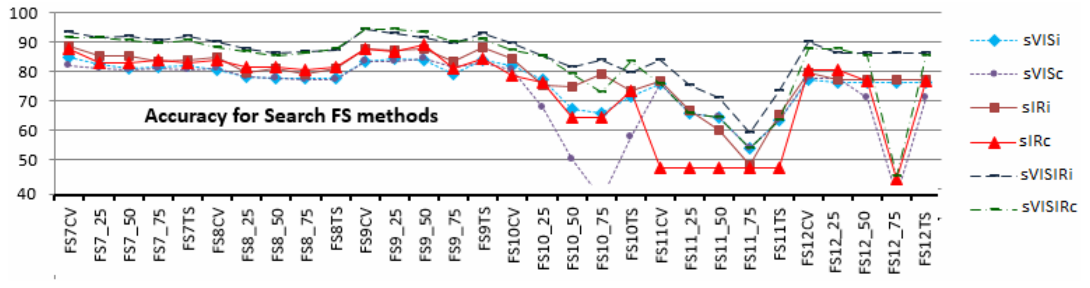


Figure 9. KNN accuracy for the Search type of FS methods (FS7…FS12)

VIII. ACKNOWLEDGMENT

The author would like to express the gratitude for having the possibility to use the Tetravision dataset provided by the VisLab and wants to address many thanks to Mr. Prof. Alberto Broggi, but also for all the collaborations with INSA in developing some of the presented results.

REFERENCES

[1] Apătean, A., “Designing Efficient Multimodal Classification Systems Based on Features and SVM Kernels Selection”, Acta Technica Napocensis, vol.57, no. 3/2016, pp.1-9, 2016

[2] Apătean, A., Rogozan, A., Bensrhair, A., “Visible-Infrared Fusion Schemes for Road Obstacle Classification”, Journal of Transportation Research Part C: Emerging Technologies, Vol. 35, pp. 180-192, 2013.

[3] Weka – Data mining software in Java, [Online] http://www.cs.waikato.ac.nz/ml/weka/, [accessed: May 1, 2016].

[4] Bouckaert, R. R., Frank E., Hall, M., Kirkby, R., Reutemann, P., Seewald, A., Scuse, D., WEKA Manual for Version 3-7-8, 2013, [accessed: Dec. 2015], http://statweb.stanford.edu/~lpekelis/13_datafest_cart /WEKA Manual-3-7-8.pdf

[5] I. Witten, et al. *Data Mining: Practical Machine Learning Tools and Machines*, 3rd ed., Elsevier, MK, 2011.

[6] LibSVM tool, https://www.csie.ntu.edu.tw/~cjlin/libsvm/, [accessed: Dec.10, 2016]

[7] Bertozzi, M., et al., "Low level pedestrian detection by means of visible and far infra-red tetravision”,*IEEE IV Symp.*, pp.231–236, 2006.

[8] Bertozzi,M., et al., "Multi stereo-based pedestrian detection by daylight and far-infrared camera, *Hammoud, R.,AugmentedVision Perception in Infrared:Algorithms and Applied Systems*, Springer Inc., pp. 371–401 (Ch16), 2009.

[9] Florea, F., *Annotation automatique d'images à partir de leur contenu visuel et des régions textuelles associées. Application dans le contexte d'un catalogue de santé en ligne*, Ph. D. Thesis, INSA, Rouen, France, 2007.

[10] Hall, M., Correlation-based Feature Selection for ML. Ph.D. Thesis, Univ. of Waikato, CS Department, Hamilton, New Zealand, 1999.

[11] Apătean, A., Rogozan, A., Bensrhair, A., “Obstacle recognition using multiple kernel in visible and infrared images”, *IEEE Intelligent Vehicle Symposium*, pp. 370-375, 2009

[12] Apătean, A., Rusu, C., Rogozan, A., Bensrhair, A., “Visible-infrared fusion in the frame of an obstacle recognition system”, 2010 IEEE Int. Conf. Automation Quality and Testing Robotics (AQTR), pp.1-6, 2010.

[13] Apătean, A., Rogozan, A., Bensrhair, “Information Fusion for Obstacle Recognition in Visible and Infrared Images”, *IEEE Int. Symposium on Signal, Circuits and Systems*, pp. 1-4, 2009.

[14] Tilca, M., Bojor, M. "Comparative Study on Different Types of Regression Applied to Unemployment in Maramures County of Romania.", Studia Universitatis „Vasile Goldis” Arad–Econ.Series26.3, pp.44-61, 2016.