



Departament de
Teoria del Senyal
i Comunicacions



Generalitat de Catalunya
**Corporació Catalana
de Mitjans Audiovisuals**



Rich Internet Application for Semi-Automatic Annotation of Semantic Shots on Keyframes

by Elisabet Carcel Folch

Directed by Xavier Giró and Xavier Vives

Escola d'Enginyeria de Terrassa (EET) - Spring 2011

Contents

1. Introduction	10
2. Requirements	12
2.1. Graphical User Interface	12
2.2. Semantic Shot detection	14
3. Working plan	17
I. Semantic shot detection	18
4. State of the art	19
4.1. Pattern recognition	19
4.2. Shot size and semantic shot detectors	21
4.3. Evaluation measures	25
5. Design	28
5.1. Classifier architecture	28
5.2. Classes definition	29
6. Development	31
6.1. Development environment	31
6.2. Features extraction	34
6.3. Annotation	35
6.4. Trainer	36
6.5. Detector	36
6.6. Dataset partition	37
7. Evaluation	39
7.1. Developed measures	39
7.2. Evaluation methodology	42
7.3. Results	43

II. User Interface	51
8. State of the art	52
8.1. iPhoto	52
8.2. GAMERA	54
8.3. Lookapp	55
8.4. IM3I	57
8.5. GAT	58
9. Design	60
9.1. Functionalities and layout needs	60
9.2. Work flow	63
10. Development	65
10.1. Development environment	65
10.2. Communications	67
10.3. Data structure	73
10.4. Features development	74
11. Evaluation	86
11.1. GUI results	86
III. Conclusions	88
12. Compliance of the requirements	89
12.1. Classifier	89
12.2. Graphical User Interface	90
13. Further work	91
13.1. Evoluting to an intelligent indexing tool	92
A. Classifier parameters analysis	95
A.1. Minimum score	95
A.2. Minimum number of elements and maximum radius	96
B. Contributions to Bitsearch blog	98
B.1. Automatic video shot type identification by Wang et al	98
B.2. Supervised, unsupervised and semi-supervised learning	99
B.3. Measures out of a confusion matrix	101
B.4. Web interface for shot type detection	103
B.5. Dataset generation	106

List of Figures

1.1. Semantic shot concept	11
2.1. Classification system requirements	12
2.2. Digitation features	13
2.3. Digitation on the browser	14
2.4. Soccer shot types	15
2.5. Parliament shot types	16
3.1. Weekly work schedule	17
4.1. Pattern recognition system based on a binary classifier	20
4.2. Training data in different types of learning	21
4.3. Classification error rate depending on the classifier used	22
4.4. Shot classes based on experimental observation	23
4.5. Proposed scheme for semantic shot classification	24
5.1. Training architecture	28
5.2. Detection architecture	29
6.1. Logos from the tools used during development	31
6.2. Screenshot showing semantic shot annotation with GAT	33
6.3. ColorLayout sample	34
6.4. Trainset and testset partition	37
6.5. One iteration of a 3-Fold Cross-Validation	38
7.1. Soccer match F1 measures for 0.7 minimum score	44
7.2. Precision, recall and F1 measure bar graph for each soccer match class	45
7.3. Soccer match detection errors	46
7.4. Catalan Parliament F1 measures for 0.9 minimum score	48
7.5. Precision, recall and F1 measure bar graph for each Catalan Parliament class	49
7.6. Catalan Parliament detection errors	50
8.1. iPhoto labeling	52
8.2. iPhoto validation	53

List of Figures

8.3. Gamera interface screenshot for Greek characters annotation	54
8.4. Retrieval analysis of the query “Sagrada Familia” screenshot	55
8.5. “Sagrada Familia” retrieval analysis screenshot	56
8.6. IM3I semi-automatic-annotations screenshot	57
8.7. Screenshot showing the positive instances of a semantic class	59
9.1. First layout proposal	61
9.2. Second layout proposal	62
9.3. Definitive layout semantic shot view	63
9.4. Interface work flow diagram	64
10.1. Eclipse URL generation for development mode	65
10.2. Google Chrome developer mode	66
10.3. Google Chrome developer plug-in	67
10.4. Domain drop box	75
10.5. AssetID text box and search button	75
10.6. Keyframes preview	76
10.7. Error message: the requested asset is not ingested in the database	76
10.8. Keyframe dragging	79
10.9. Minimum score drop box	79
10.10Sort button	80
10.11Unsorted close-up semantic shot	80
10.12Sorted close-up semantic shot	80
10.13Page validation button	81
10.14Validated page	82
10.15Validated semantic shot	82
10.16Semantic shot keyframes panel	83
10.17Quality keyframes annotation	83
10.18Pagination menu	84
10.19Progress bar showing different percentages	85
10.20Tooltip showing the keyframe’s title and score	85
11.1. GUI layout	87
A.1. 0.1 vs 0.8 minimum score on Socer matches classifier	95
A.2. 0.9 minimum score on Socer matches classifier	96
A.3. Maximum radius ribbons at 0.7 minimum score	96

List of Tables

4.1. Perfect confusion matrix performance	25
4.2. Confusion matrix example	25
4.3. Confusion matrix for binary classifiers	26
5.1. Classes defined for the soccer match ontology	30
5.2. Classes defined for the Catalan Parliament ontology	30
6.1. Used Javadoc tags	32
6.2. Labeled instances for the soccer match ontology	35
6.3. Labeled instances for the Catalan Parliament ontology	35
7.1. ConfusionMatrix methods	39
7.2. Confusion matrix for multiclass classifiers	40
7.3. Best soccer match confusion matrix instances	45
7.4. Best Catalan Parliament confusion matrix	49
10.1. Keyframe class variables	73
10.2. KeyframeVectors class variables	74
10.3. KeyframeVectors Methods	74
10.4. DragAndDrop class variables	77
10.5. DragAndDrop Methods	78

Acknowledgements

I thank Professor Xavier Giró for his useful advice, paper corrections and opinions in our numerous meetings.

I thank Xavier Vives, my advisor at the CCMA, and his team for supporting me during this four months.

I also thank Manel for all the time he spent in this project and his friendly support.

Abstract

This thesis describes the graphical user interface developed for semi-automatic keyframe-based semantic shot annotation and the semantic shot classifiers built. The graphical user interface aims to optimize the current indexation process by substituting manual annotation for automatic annotation and validation.

The system is based on supervised learning binary classifiers and web services. The graphical user interface provides the necessary tools to fix and validate the automatic detections and to learn from the user feedback to retrain the system and improve it.

Results of the classifiers evaluation, performed using cross-validation methods, show a good performance in terms of precision and recall. The graphical user interface has been described as complete and easy to use by a professional documentalist at a broadcast company.

1. Introduction

This is a cooperation project between the Technical University of Catalonia (UPC) and the Catalan Broadcasting Corporation (CCMA). The CCMA¹ offers a public broadcasting service promoting the Catalan language and culture. The corporation is organized in several audiovisual companies: Televisió de Catalunya (TV3), Catalunya Ràdio, CCRTV Interactiva, CCRTV ASI and Activa Multimèdia Digital.

Companies related to audiovisual production have to deal with an incredible amount of incoming data which has to be properly indexed into their repositories for retrieval purposes. Nowadays the CCMA's digital archive has stored up to 135.000 hours of video with annual growth of about 25.000 hours. From these 135.000 stored hours only about 75 are daily retrieved for several purposes such as reusing fragments for summaries, content selling or dubbing. As daily retrieved information involves only 0.06% of the total data, narrowing down the search of video content by getting specific results is still a demanding issue to suit the Documentation Department needs.

The Image Processing Group at the UPC has already worked with the company providing prototype tools to be integrated into the company's own content management system, the Digitium, in order to easier the retrieve process and to extend it to multimodal searches[1, 2, 3].

In like manner, this bachelor thesis is aimed at creating a web interface for keyframe-based semantic shot annotation. A semantic shot provides information about the camera shot size in terms of field of view (long shot, medium shot, clouse-up...) plus a subject in the scene, as for example a player, the court or the audience in a soccer match. Image 1.1 shows a visual representation of the semantic shot concept.

¹<http://www.ccma.cat>

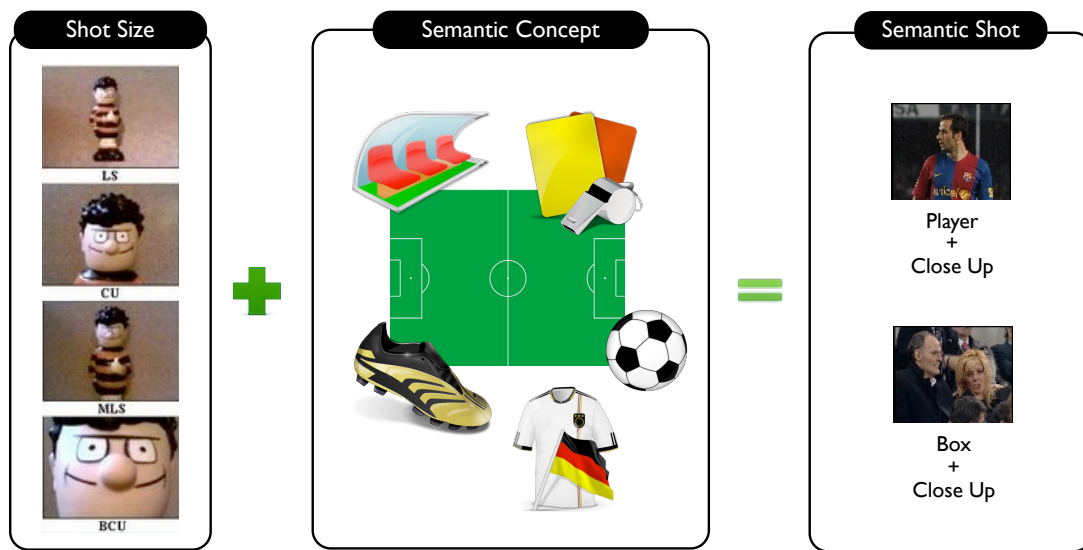


Figure 1.1.: Semantic shot concept

Nowadays, semantic shot information is manually annotated by documentalists as a storyboard to explain what is going on in a video sequence. The new tool will reduce the time taken to generate semantic shot metadata by substituting manual annotation for automatic annotation and validation. This metadata will also provide further accuracy in video retrieval when using it along with other semantic concepts in the textual search.

This work extends a research line developed at the Image Processing Group at the UPC, which started with the implementation of C and C++ engines for feature extraction[15, 12] used for a semantic shot detector based on k-NN classifier[17] and in graphical annotation tool for classifier evaluation[13].

2. Requirements

The CCMA demands a semantic shot detector to automatically generate new labels for the metadata. The results have to be supervised by a documentalist, validating the detections or correcting them if necessary. Moreover, the system has to learn from the user feedback allowing to train the system and improve it. To do so, a graphical user interface is needed.

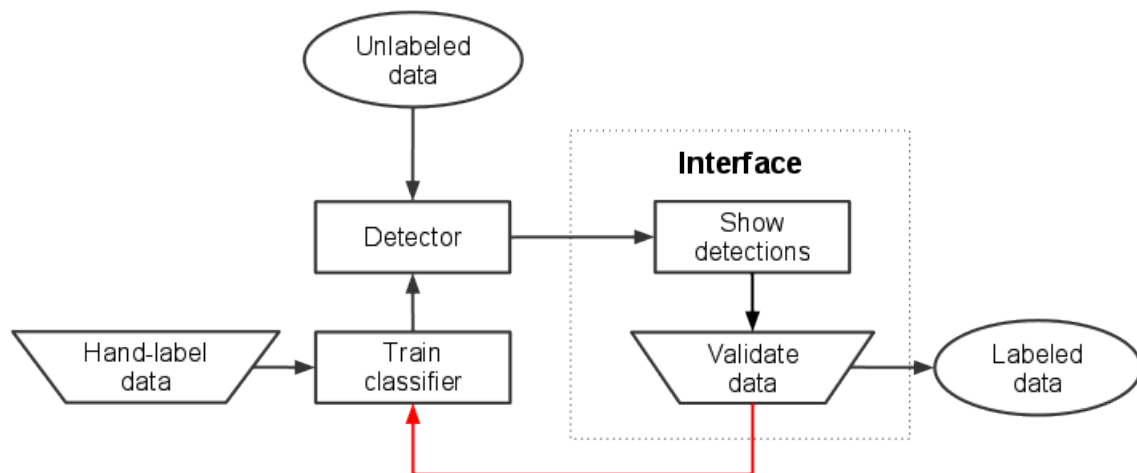


Figure 2.1.: Classification system requirements

2.1. Graphical User Interface

For a given domain and an asset (group of keyframes of that domain) the interface will have to show the automatic shot type identification and provide the necessary tools for validating the automatic detections or correcting them if necessary.

When defining the specific needs for the GUI there were several conditions that the project had to accomplish in order to ease the integration with the existent software in the company: the DigionSuite.

2.1.1. DigionSuite

The DigionSuite[4] is the digital archiving and asset management system developed by Activa3¹, which is one of the CCMA's companies.

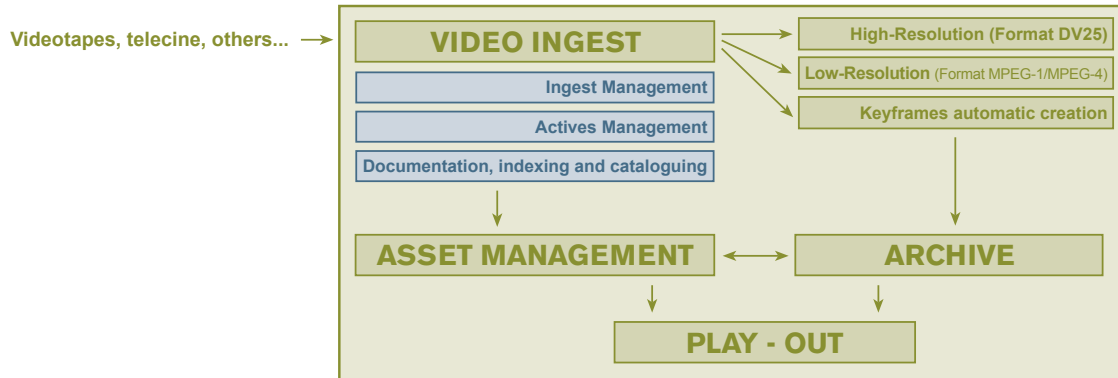


Figure 2.2.: Digion features

DigionSuite is able to ingest contents from different sources: videotapes, links, etc. The content ingest is performed simultaneously in low-resolution (MPEG-1 format at 1.2Mb/s) and broadcast resolution (DVCPPro, MPEG-2, DV, DVCam), which allows viewing of either all or part of the content from any point in a computer network while ensuring excellent quality for professional editing and broadcast.

This system also catalogues every shot change in real time using keyframes. This makes it fast and easy to track down content and it is an essential feature for the development of this project, as it provides the images to work with.

DigionSuite includes tools to catalogue, index-link, search, automatically copy, delete and store assets so that users can quickly find, consult and view stored content either simultaneously or individually. Content search, location and retrieval can also be performed through a storyboard created during the ingest process and based on each shot change.

2.1.2. Interface requirements

The story board section of the DigionSuite is where the semi-automatic annotation will be in the future. As the Digion has no semi-automatic annotation section, a new window for annotation and training purposes has to be created and placed in the indexing section.

¹<http://www.activa3.net>

2. Requirements

The interface has to be flexible so the final user can adjust the sensibility threshold of the system for validation tasks depending on the results. Other requirements are that it has to be generic, as in the future they want to be able to train new models for detection besides semantic shots.

The Digition is based on open software and its accessible through a web browser. To make the GUI completely integrable into the Digition the programming language to be used is Java Google Web Toolkit² (GWT), a development toolkit for building complex browser-based applications.



Figure 2.3.: Digition on the browser

2.2. Semantic Shot detection

One requirement of the semantic shot classification is that it has to be developed using the binary classification tools that the Image Processing Group has developed.

²<http://code.google.com/intl/es-ES/webtoolkit/>

The system has to be able to distinguish between different semantic shots. This has to be done depending on the video topic, as the basics of the visual language and the subjects of interest may differ from one domain to another.

All audiovisual content at the CCMA is organized in assets. An asset includes a video sequence, keyframes with its specific time code, general information such as title or description of the content and some annotations about specific events in the sequence. The domain information is assumed to be already indexed in the asset metadata. The Documentation Department at the CCMA has defined two different domains of interest: the Catalan Parliament and soccer matches.

2.2.1. Soccer matches

In soccer matches most of the keyframes where the playground is shown are not of interest. These general shots have to be differentiated from those which are of interest such as the audience or the stadium overview. Other semantic shots to detect are close-ups showing a specific player or people in the audience or the box.

These are the listed semantic shots for the soccer match domain: player medium shot, player close up, stadium overview, audience, banner and box. Figure 2.4 shows two keyframe examples for each semantic shot.

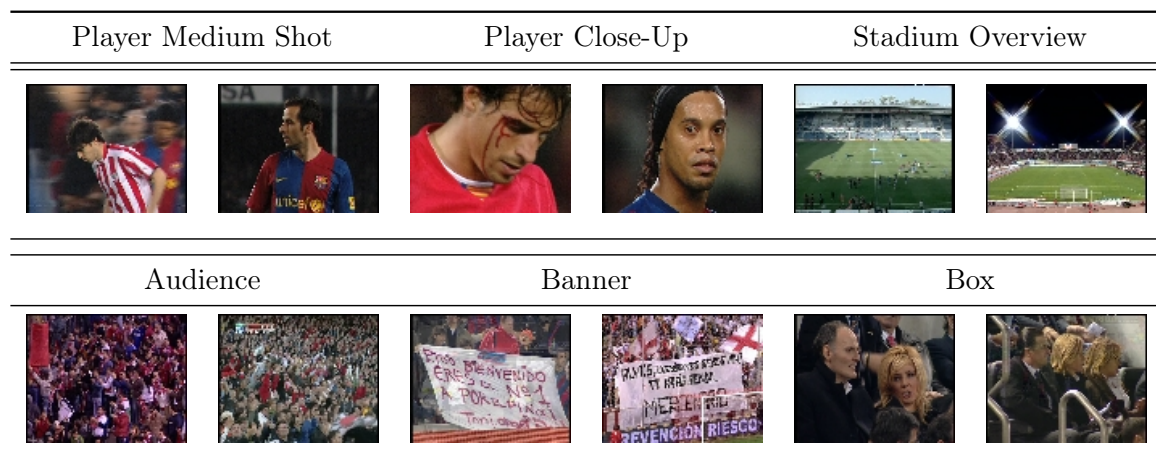


Figure 2.4.: Soccer shot types

2. Requirements

2.2.2. Catalan Parliament

The parliament is a controlled environment where camera location and shot sizes remain the same. This makes it easy to extract event information out of the shot size used in each keyframe.

The subjects of interest in the Parliament are the chamber, the bureau, the parliamentarians and the president of the Parliament. The semantic shots to be detected in the Parliament domain are: president medium close up, medium close up, medium shot, general Chamber shot and general Bureau shot. Figure 2.5 shows two keyframe examples for each semantic shot.

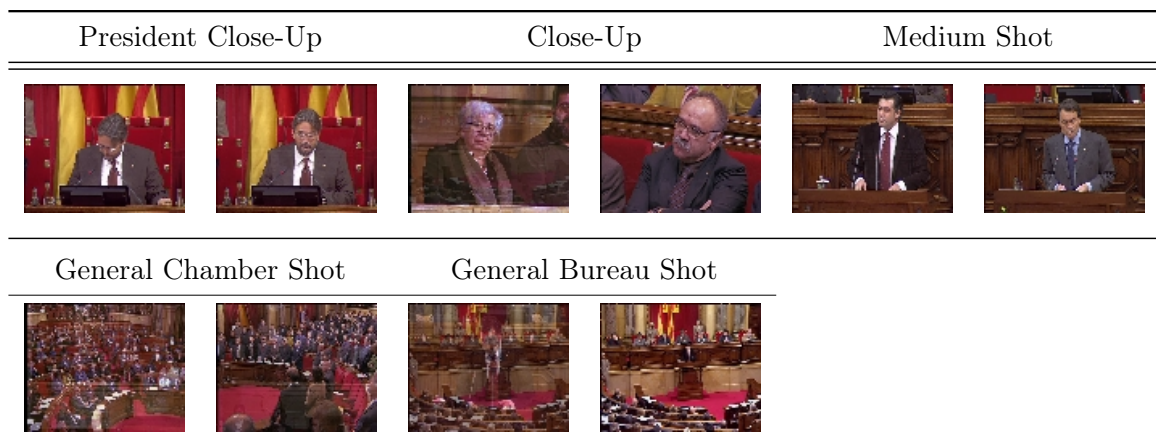


Figure 2.5.: Parliament shot types

3. Working plan

	February				March				April				May				June			
Tasks	1st	2nd	3rd	4th	1st	2nd	3rd	4th	1st	2nd	3rd	4th	1st	2nd	3rd	4th	1st	2nd	3rd	4th
UPC																				
Classification state of the art	■	■	■																	
Environment familiarization	■																			
Hand-label annotation		■	■																	
Experiment development				■	■	■	■	■												
Director meeting	■	■	■	■	■	■	■	■												
CCMA																				
Labeling interfaces state of the art					■	■														
Environment familiarization					■															
Communications configuration																				
GWT learning demos					■															
Inicial proposals						■	■	■												
Selected design development									■	■	■	■								
Design refinement													■	■	■	■				
Client meeting									■				■				■			
Demo recording																	■			
Thesis																				
Latex familiarization									■											
Preamble configuration										■	■	■								
Thesis redaction										■	■	■	■	■	■	■	■			

Figure 3.1.: Weekly work schedule

Part I.

Semantic shot detection

4. State of the art

The study of automatic systems for audiovisual data based on its content has been a growing demand these last years. These systems are based on pattern recognition techniques that will be introduced in this chapter for the correct understanding of the actual system. The main metrics used to evaluate classifiers performance are also discussed.

4.1. Pattern recognition

Pattern recognition consists on automatically assigning a semantic label or class to an input instance given its features. This process can be divided into data acquisition, feature extraction and classification of the data based on features.

For pattern recognition of video images data acquisition consists on generating keyframes out of the video. The keyframe generation can be done periodically by the timestamp, when there is a change in the scene or when an specific event occurs, depending on the needs and possibilities.

Several features can be extracted from the keyframes. These features are called visual descriptors and can be divided in two main groups: general information descriptors (color, texture, shape, motion) which are low level descriptor and specific domain information descriptors (face detection or recognition).

Once a feature is associated with a keyframe, the system can proceed to classify it. A classification problem consists of two stages: training and detection. For training a classifier, instances of a semantic concept are manually annotated so their vectors of features can be processed to generate a model for the detector. The detector compares the model with the unlabelled instances and outputs a probability score for every modelled class.

4. State of the art

Figure 4.1 shows a basic scheme of a binary classifier where parliament keyframes have been labeled as positive and negative instances and which features have been extracted in order to train the classifier. The model generated by the trainer is then used to compute the score of unlabeled data features. This example adds a decision stage, where a threshold determines the minimum score needed at detection.

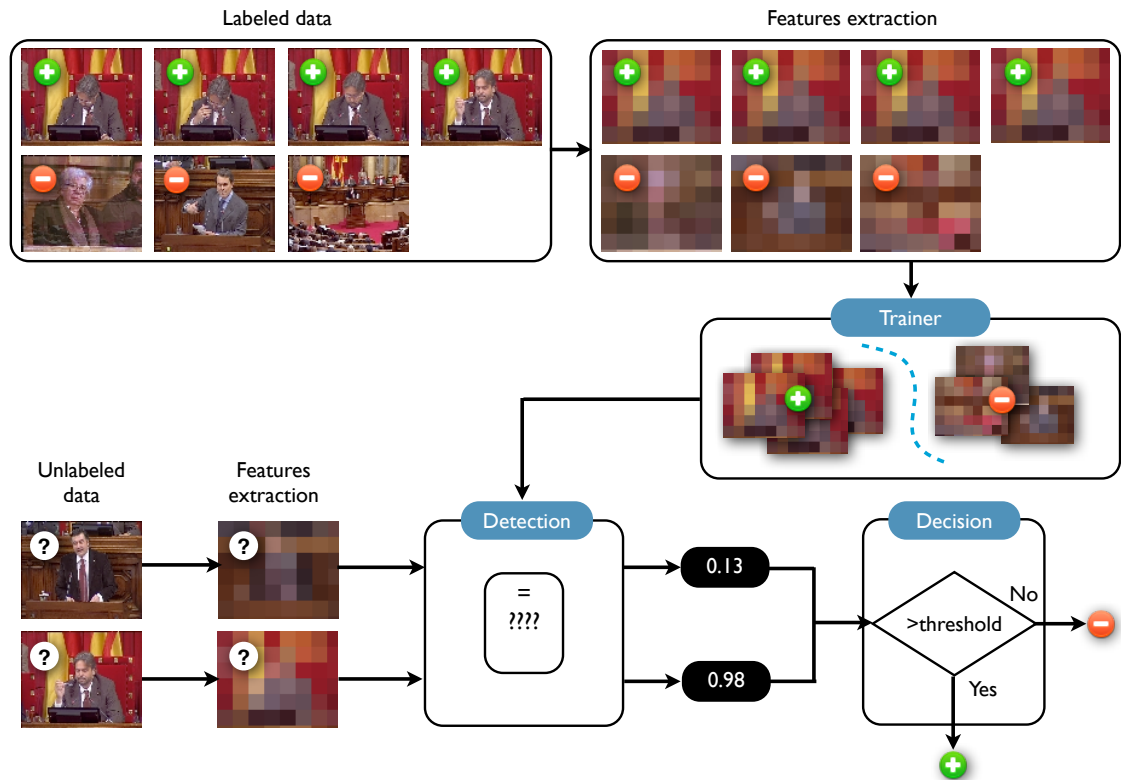


Figure 4.1.: Pattern recognition system based on a binary classifier

4.1.1. Supervised, unsupervised and semi-supervised learning

Pattern recognition can use different machine learning techniques. The main machine learning categories are: supervised learning, unsupervised learning and semi-supervised learning. The main difference between them is whether the training data has been hand-labeled or not.

Supervised learning assumes that a set of training data has been provided, consisting of a collection of instances that have been manually labeled with the expected values.

On the other hand, unsupervised learning attempts to automatically find similar patterns in unlabeled data.

A combination of both supervised and unsupervised learning results in semi-supervised learning which uses a combination of labeled and unlabeled data as training data. The classifiers iteratively update the models by progressively annotating the unlabeled data with the already labeled data.

Figure 4.2 shows a diagram explaining the procedure of training data depending on the learning type.

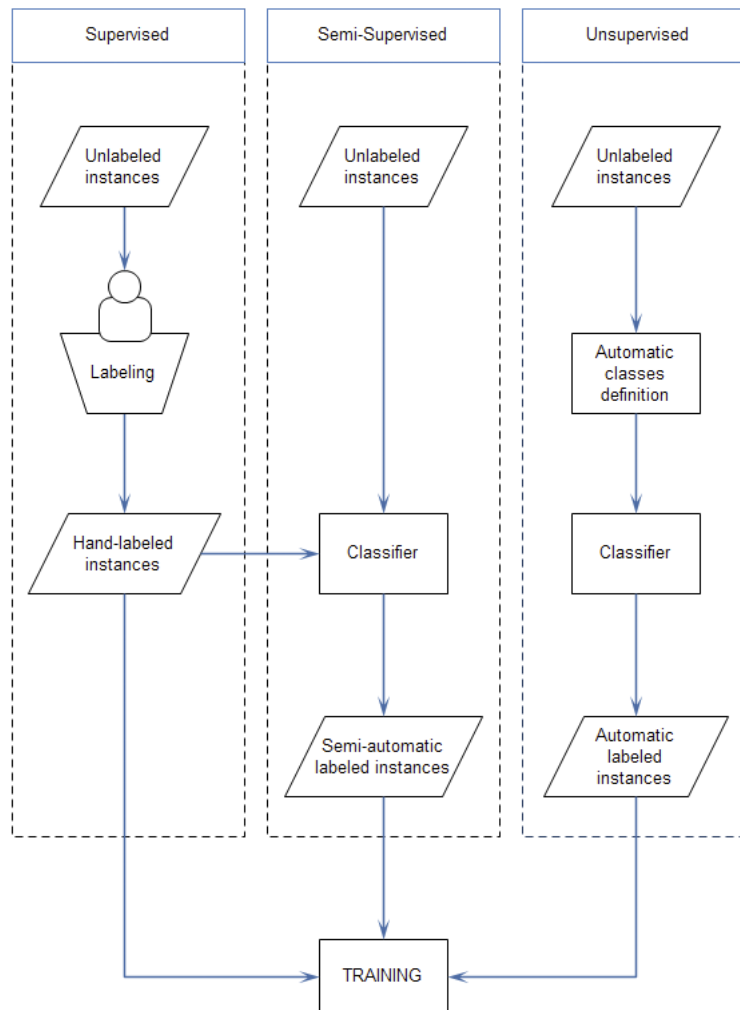


Figure 4.2.: Training data in different types of learning

4.2. Shot size and semantic shot detectors

Previous work has addressed shot size and semantic shot classification. This chapter provides an overview of the techniques developed for video analysis based on multiclass classification.

4.2.1. An Efficient Automatic Video Shot Size Annotation Scheme by Wang et al

This work describes a shot-type recognition system based on three possible categories: Close-up, Medium shot and Long shot[8].

Two feature vectors are created, one corresponding to global features such as color, edge and texture and the other one to local features involving regions and its features. Semi-supervised learning is used in order to obtain valid data to train the system.

Each vector of features is sent to a different classifier (global and local) to decide which shot size suits best the image among the three existent classes. A combination of both classifiers' results is used in order to improve its efficiency. Cost-sensitive decisions for misclassifications are calculated to provide a better cost minimization based classifier.

Tests are made using 20 hours of video extracted from home video camcorders and then divided into 1000 shots according to timestamps. As in this stage some shots may include frames of different shot sizes, each shot is divided into 4000 sub-shots, assuming now identical shot size within a sub-shot.

All detection results are generated as the average of 10 runs, where a run is an iteration done by using 20% of the samples randomly selected as training data and the other 80% to test the system.

Figure 4.3 shows how global descriptors work better in shot type identification, though it can be pursued a better performance using a combined low level and mid level decision.

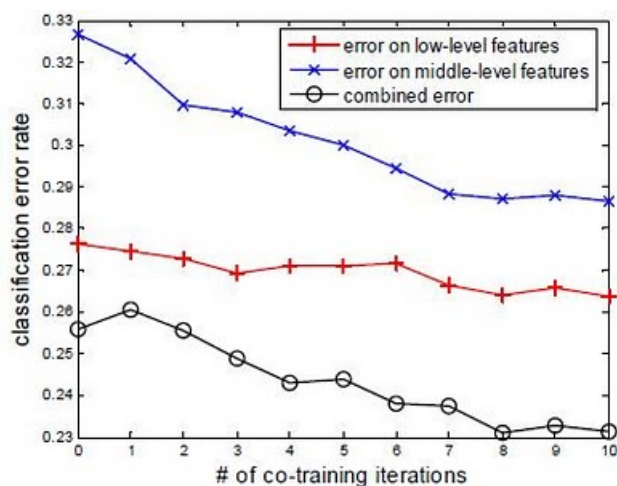


Figure 4.3.: Classification error rate depending on the classifier used

4.2.2. A Unified Framework for Semantic Shot Representation of Sports Video by Duan Xu et al

This paper presents a frame work for semantic shot classication of sports video[9]. The proposed scheme has predefined a number of semantic shot classes for tennis, soccer and basketball with a relevant semantic meaning and which cover most of the sports broadcasting video.

A detailed table of the defined classes and its characteristics such as semantic meaning and percentage of ratio among the other classes is shown in figure 4.4.

Type	Class	Ratio (%)	Major Class (Tick Besides)	Potential Semantic Linkage
Tennis	Close-up	37.5	√	Break, Serve
	Court View	29.8	√	Play
	Player Medium-View	16.8	√	Break
	Audience	9.2		Active Reaction
	Bird-View	1.0		Begin a new game
	Replay	4.3		Exciting moment
	Undefined	1.4		
Soccer	Close-up	26.0	√	Break, Throw-in
	Field View	33.6	√	Play
	Following	22.1	√	Play
	Player Medium Still	3.0		Free Kick, Penalty Kick
	Audience	0.7		Active Reaction
	Corner Kick	0.7		
	Goal View	4.8		Offense
	Replay	4.1		Exciting moment, Foul
Undefined	5.0			
Basketball	Close-up	41.9	√	Break, Penalty
	Fast Break	22.6	√	Play
	Court View Still	5.4		Begin a new game
	Player Medium View	4.3		Highlight
	Penalty	7.5		
	Audience	7.5		Active Reaction
	Bird-View	2.2		Begin or End a game
	Replay	7.5		Exciting Moment
Undefined	1.1			

Figure 4.4.: Shot classes based on experimental observation

4. State of the art

Note that each domain has an “Undefined” class, also known as clutter class (as these kind of classes are referred in this project). Having a clutter class means the classifier not only has to identify instances belonging to the defined classes but also identify instances that do not belong to any of the classes.

As in our approach, the proposed scheme makes use of domain knowledge (in this case the specific sport among tennis, soccer or basketball) to perform the video shot classification. For a given domain they construct middle level features (camera motion patterns, motion entropy, shot pace, active regions etc) from low level features.

The relation between low-level and middle-level is defined by supervised learning methods. Once the middle-level shot attributes are defined, video shots are classified into the predefined video shot classes using Bayesian classifiers and support vector machines into their defined semantic shots. The whole process is represented in figure 4.5.

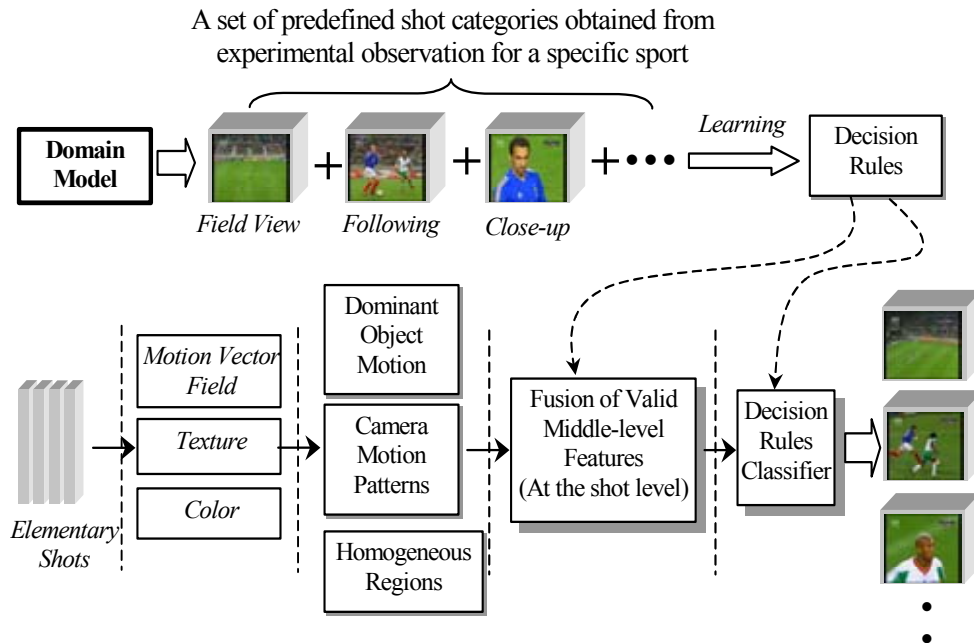


Figure 4.5.: Proposed scheme for semantic shot classification

The system has been build under the Microsoft DirectShow framework, using it for video processing, visualization and flexible video streaming. In terms of performance, the system achieved a 80~95% of precision.

4.3. Evaluation measures

This section explains the most common measures to evaluate a classifier. One of them, the confusion matrix, is aimed at evaluating multiclass classifiers while the other ones are commonly used to evaluate binary classifiers.

4.3.1. Confusion matrix

A confusion matrix is used in supervised learning for comparing the outcome classification of an item with the desired classification. Each row of the matrix represents the instances that have been automatically predicted in a class, while each column represents the hand-labeled instances in a class. In table 4.1 a perfect confusion matrix is shown in terms of percentages, 100% of the predicted instances belong to the actual class.

		Automatic		
		class1	class2	class3
Manual	class1	100%	0%	0%
	class2	0%	100%	0%
	class3	0%	0%	100%

Table 4.1.: Perfect confusion matrix performance

This matrix is useful for observing which classes have miss-identified items as other classes. A number different to '0' out of the confusion matrix diagonal would be a mistake. In table 4.2 only twelve instances out of fifteen are labeled as class1, while the other three are labeled incorrectly as class2. It has also mistaken one of the fourteen instances from class2 to belong to class1. Class3 works fine with 0 mistaken instances.

		Automatic		
		class1	class2	class3
Manual	class1	12	1	0
	class2	3	13	0
	class3	0	0	20

Table 4.2.: Confusion matrix example

4.3.2. Precision and Recall

Precision or specificity is a measure of the ability of a system to present only relevant instances. It measures the exactness or fidelity of the system.

$$Precision = \frac{\text{amount of correct instances detected}}{\text{total amount of instances detected}} \quad (4.1)$$

Recall or sensitivity is a measure of the ability of a system to present all relevant instances, so it is used for evaluating the completeness of results.

$$Recall = \frac{\text{amount of correct instances detected}}{\text{amount of instances in the collection}} \quad (4.2)$$

In predictive analysis a two row and two column confusion table is commonly used for evaluating binary classifiers. It reports the number of true negatives, false positives, false negatives, and true positives. True negatives and true positives are properly classified items, while false positives and false negatives occur when obtained classification do not match correct classification. This is illustrated in table 4.3.

		Automatic	
		positive	negative
Manual	positive	true positive <i>tp</i>	false positive <i>fp</i>
	negative	false negative <i>fn</i>	true negative <i>tn</i>

Table 4.3.: Confusion matrix for binary classifiers

This table is used to perform precision and recall metrics. Now equations 4.1 and 4.2 can be defined as:

$$Precision = \frac{tp}{tp + fp} \quad (4.3)$$

$$Recall = \frac{tp}{tp + fn} \quad (4.4)$$

It is difficult to compare different systems in terms of precision and recall, as both measures are independent. Moreover, when recall increases precision tends to decrease: as more relevant instances are retrieved, the more non-relevant instances are retrieved.

4.3.3. F1 and F β measure

F-measure considers both precision and recall providing a single measurement for a system avoiding having two independent measures.

$$F = 2 \cdot \frac{\textit{precision} \cdot \textit{recall}}{\textit{precision} + \textit{recall}} \quad (4.5)$$

In order to give different weights to precision and recall, the F-measure was derived so that F β measures the effectiveness of retrieval with respect to a user who attaches β times as much importance to recall as precision[8].

$$F_{\beta} = (1 + \beta^2) \cdot \frac{\textit{precision} \cdot \textit{recall}}{(\beta^2 \cdot \textit{precision}) + \textit{recall}} \quad (4.6)$$

5. Design

There are two kinds of classifiers: binary and multiclass classifiers. Binary classifiers decide whether a feature is present or not while multiclass classifiers deal with more than one class rather than having a concept or not.

As semantic shot classification needs multiclass detection and the Image Processing Group provides binary classification tools, a multiclass classifier has been implemented combining binary classifiers. The strategy used has been training a binary classifier for each semantic shot and then returning the class which has obtained the highest score.

5.1. Classifier architecture

5.1.1. Training stage

The training stage involves labeling positive and negative instances of each semantic shot and extract visual descriptors to train as many classifiers as semantic shots (but the clutter class) so one model per class is generated. The whole process can be seen on Figure 5.1.

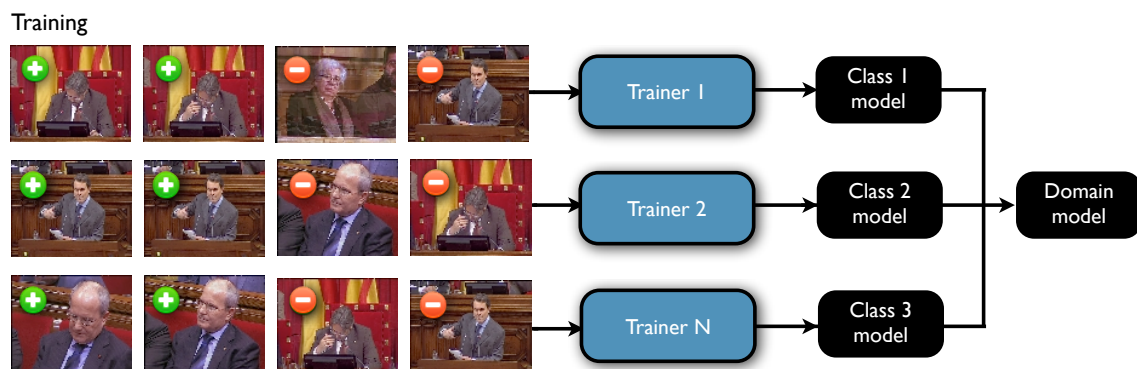


Figure 5.1.: Training architecture

5.1.2. Detection stage

At detection stage, for a given set of visual descriptors of an unlabeled instance, each semantic shot classifier would output a score using the models generated at the training stage. As each instance can only belong to one of the classes the maximum score among all classifiers corresponds to the most probable semantic shot.

The detection stage also includes a decision stage, where the classifier decides whether an instance belongs to the class which has obtained a higher score or to the clutter class depending on a threshold. This threshold is essential as instances that are not of interest or do not belong to any of the classes have to be classified as clutter.

The detection procedure can be seen on Figure 5.2

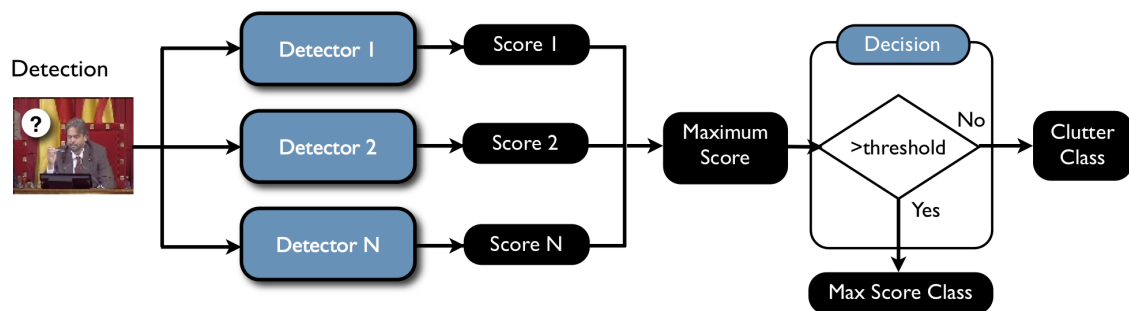


Figure 5.2.: Detection architecture

5.2. Classes definition

An ontology is a representation of a set of concepts (classes) within a domain. In order to annotate positive and negative instances for semantic shot detections at soccer matches and the parliament domains, two different ontologies need to be defined with their respective semantic shots. The ontologies defined for soccer matches and the Catalan Parliament and their respective classes are listed in Table 5.1 and 5.2 respectively.

Soccer match Ontology	
Class 1	Player Medium Shot
Class 2	Player Close-Up
Class 3	Stadium Overview
Class 4	Audience
Class 5	Banner
Class 6	Box
Class 7	Clutter

Table 5.1.: Classes defined for the soccer match ontology

Catalan Parliament Ontology	
Class 1	President Close-Up
Class 2	Close-Up
Class 3	Medium Shot
Class 4	General Chamber Shot
Class 5	General Bureau Shot
Class 6	Clutter

Table 5.2.: Classes defined for the Catalan Parliament ontology

As it can be seen, the classes defined for each domain are the ones stated at the requirements plus an additional one corresponding to all the other shot-types that the CCMA does not want to detect as any of the classes.

The clutter class containing all instances that are not of interest was not included at first because the idea of generating a model out of images with no visual similarities between them was desestimated.

The problem of not including it in the ontology is that the lack of annotations on this class does not allow the estimation of the classifier model performance given an instance that does not belong to any of the classes. In like manner, some parameters like the minimum score at the decision stage had to be set depending on the evaluation of this class, so the clutter class is included in the ontology, so instances can be labeled as clutter class to be used at detection but not by the trainer.

6. Development

This chapter focuses on the development of the classifier, including the development environment and all the steps necessary to build the classifier.

6.1. Development environment

This section briefly describes the technologies and programming tools used for the classifier development and evaluation.



Figure 6.1.: Logos from the tools used during development

6.1.1. Eclipse

Eclipse¹ is a development environment open source platform programmed mainly in Java. It allows developing projects in Java, C, C + +, Python PHP and many others, providing the necessary connectors for each programming language. For the classifier development Eclipse has been used as an IDE.

6.1.2. Subversion

Subversion² is an open source software configuration management (SCM) tool. This version control system is integrated in the UPSeek Eclipse project with the Subclipse

¹<http://www.eclipse.org/>

²<http://subversion.tigris.org/>

6. Development

connector. Each member of the UPSeek has a branch to develop their code. All branches are associated to a unique trunk that contains the shared version of the project.

These are the Subversion commands that have been used through the development of the classifier:

- `svn commit filename`: checks-in local file, files or directory into the repository.
- `svn update`: migrates all updates from the repository to the user local copy.
- `svn-rebase`: updates the developer branch with the last trunk modification.
- `svn-deliver`: updates the trunk with the developer branch modifications

6.1.3. Java

Java³ is a programming language developed by Sun Microsystems, which is now subsidiary of Oracle Corporation. Java is a general-purpose, concurrent, class-based, object-oriented language. One of the advantage of using Java is that its applications are compiled to a class file (bytecode) that can run on any Java Virtual Machine (JVM) regardless of the computer operating system.

6.1.4. Javadoc

Javadoc⁴ is a documentation generator from Sun Microsystems for generating API documentation in HTML format from Java source code. The Javadoc format has been used for documenting Java classes and methods so other UPSeek users can use them. Some IDEs such as Eclipse automatically generate Javadoc HTML. Table 6.1 shows the used tags for documenting the developed methods.

Tag & Parameter	Usage	Applies to
@param <i>name description</i>	Describes a method parameter	Method
@return <i>description</i>	Describes the return value	Method

Table 6.1.: Used Javadoc tags

³<http://www.java.com/>

⁴<http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html>

6.1.5. MATLAB

MATLAB⁵ is a numerical computing environment developed by MathWorks. This program allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces and interfacing with programs written in other languages, including C, C++ and Java. It has been used to perform mesh plots of the experiments results.

6.1.6. GAT(Graphic Annotation Tool)

GAT⁶[14] provides an interface to create ontologies and to label positive, negative and neutral instances for each ontology class given the visual descriptors of the images. GAT generates an annotation file in XML referencing the descriptors of each labeled instance in a class.

Though GAT is capable of labeling the regions of an image or annotate video sequences, for this project each instances corresponds to a different image.

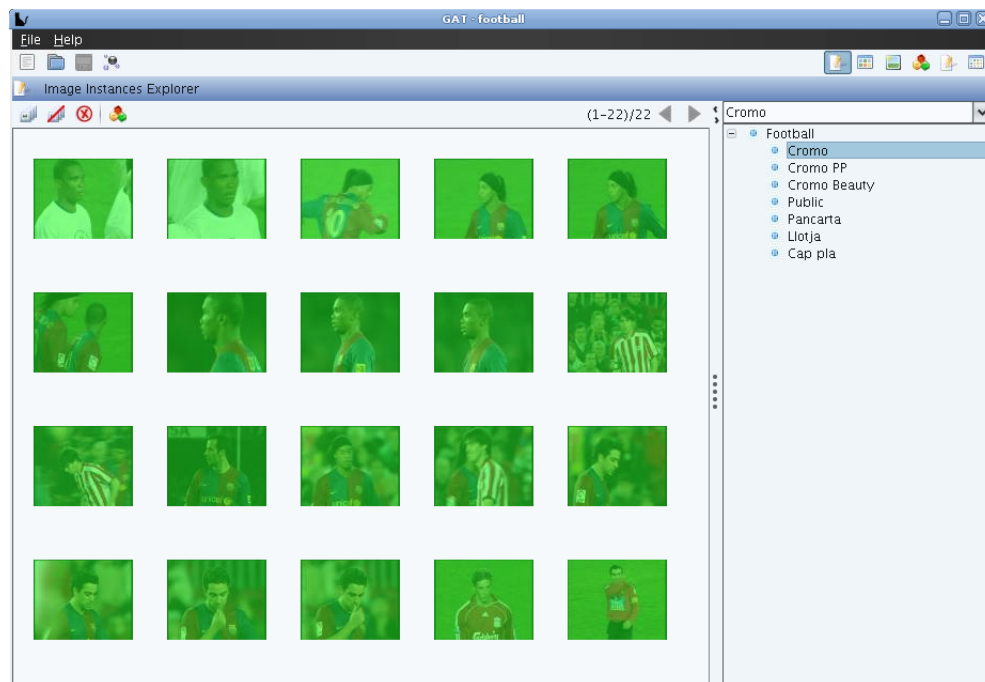


Figure 6.2.: Screenshot showing semantic shot annotation with GAT

⁵<http://www.mathworks.com/products/matlab/>

⁶<http://upseek.upc.edu/gat/>

6.2. Features extraction

The first step in a classification process is the feature extraction. For this project MPEG-7 image descriptors have been used to provide standardized descriptions of visual aspects of the keyframes[12].

These descriptors have been automatically extracted for each keyframe and then stored in a XML file. Image features have to be common for the elements contained in the same semantic class and different from the other semantic classes.

The descriptors chosen to approach shot-type detection have been the following ones:

- *Dominant Color Descriptor* (DCD): allows specification of a small number of dominant color values as well as their distribution or variance.
- *Color Structure Descriptor* (CSD): identifies localized color distributions using a small structuring window.
- *Color Layout Descriptor* (CLD): captures the spatial layout of the representative colors on a 8x8 grid superimposed on the image. Each block has the value of the most representative color within the block.

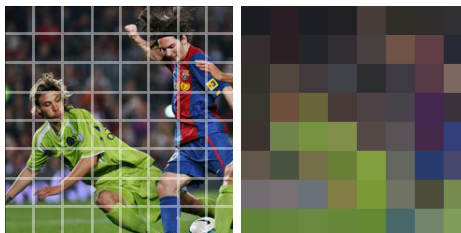


Figure 6.3.: ColorLayout sample

- *Edge Histogram Descriptor* (EHD): represents local edge distribution in an image. The image is divided into 4x4 sub-images and further divided into smaller blocks. Each block represents the value of the most representative edge, being the possible values having vertical, horizontal, diagonal or non-directional edges. Then an histogram is compute given the five possible edge types, obtaining a single histogram coefficient for each block.

6.3. Annotation

The ontologies and labeling have been done using GAT. As in semantic shot detection a keyframe can only belong to one semantic shot, each positive instance in a class has to be labeled as a negative instance of the other classes. So only positive labeling has been done with GAT, adding the negative instances further on with a program that has been developed in Java for this project.

Table 6.2 and 6.3 list the number of hand-labeled instances for each semantic shot of soccer match and Catalan Parliament.

Soccer matches positive instances	
Player Medium Shot	116
Player Close-Up	87
Stadium Overview	4
Audience	19
Banner	4
Box	6
Clutter	278
Total number of instances	514

Table 6.2.: Labeled instances for the soccer match ontology

Catalan Parliament positive instances	
President Close-Up	16
Close-Up	63
Medium Shot	68
General Chamber Shot	13
General Bureau Shot	23
Clutter	84
Total number of instances	267

Table 6.3.: Labeled instances for the Catalan Parliament ontology

GAT generates an MPEG-7 XML file for each ontology, containing its classes name. It also creates a new XML annotation file showing the different classes of the ontology referencing the positive and negative instances that had been labeled in that class. This annotation file is used by the trainer to generate a classification model.

6.4. Trainer

The program used for training the system has been the B_TRAINER⁷, which implements a clustering algorithm[3, 16] for determining which annotated instances will be used for training. It has two parameters to be set: the minimum number of elements within a cluster and the maximum radius of it.

- **Minimum number of elements:** defines the minimum elements inside the maximum radius.
- **Maximum radius:** defines the maximum distance of the cluster. Elements out of the radius boundaries are not used for the training process.

It cannot be known beforehand which parameters works best, consequently a optimum parameter search has been done through experiments.

A function that calls B_TRAINER for a minimum number of elements, maximum radius value, the ontology path and the annotation files has been developed for this project in order to generate the model to be used by the detector.

6.5. Detector

The program used for detection has been the B_DETECTOR⁸, which for a given unlabeled test dataset and the model generated by the trainer is able to detect the class with the highest score. The score is a float number compressed between 0 and 1, where 0 means null similarities with the model and 1 is the highest probability of having detected the semantic concept.

As the trainer the detector has an input parameter that has been searched through experiments:

- **Minimum Score:** determines the minimum score to consider a detection as valid. If the score of an instance is lower than the specefied minimum score, the detector leaves it undetected.

Finding the optimum minimum score is essential to properly detect instances that do not belong to any of the classes.

⁷Program developed by the Image Processing Group at the UPC

⁸Program developed by the Image Processing Group at the UPC

6.6. Dataset partition

For testing a classifier, the annotated positive and negative instances of all classes have to be split into a trainset and a testset so the trainer and the detector use different sample data for the experiment.

In this case, as there were few labeled instances, 80% of the each class sample data has been used to train its classifier while the 20% left has been used for detection.

- **Trainset:** Each positive instance is then added as a negative instance of the other classes. Note that there are no clutter positive instances in the trainset as this class is not trained.
- **Testset:** only positive instances are used for detection, including the clutter class.

Figure 6.4 show an example with 80% of each class instances used for training and 20% for testing after deleting the eight clutter class positive instances belonging to the trainset.

	Testset	Trainset																			
Class 1	1 1	1 1 1 1 1 1 1 1	2 2 2 2	- - - - - - - -																	
Class 2	2	2 2 2 2	1 1 1 1 1 1 1 1	- - - - - - - -																	
No class	- -																				

Figure 6.4.: Trainset and testset partition

There are different methods to split data into the mentioned datasets, for this project two different ways have been developed and tested: K-fold cross validation and repeated random sub-sampling.

6.6.1. Modified K-fold cross-validation

The K-fold cross-validation algorithm generates K different data subsets, making sure all data is used for both training and detection for each iteration.

As a modification of the K-fold cross-validation[13], where data are partitioned into K equal parts and the ratio between the sizes of the different subsets is not defined by the user, the parameter K has been calculated automatically to build the testset and trainset for the given trainset percentage, allowing more flexibility.

6. Development

This procedure is repeated K times and the predictions of the K dataset are averaged. The whole procedure can be done several times with aleatory data order for each iteration.

Figure 6.5 shows an example of one iteration of the K -Fold cross-validation using 3 folds.

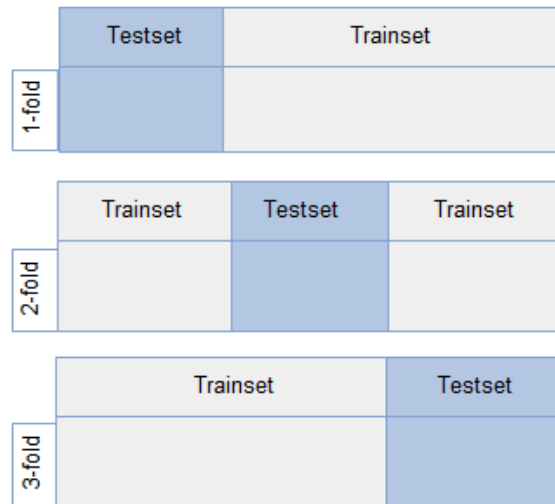


Figure 6.5.: One iteration of a 3-Fold Cross-Validation

6.6.2. Repeated random sub-sampling

Random sub-sampling consists on sorting the data in an aleatory order and then dividing it into the dataset and trainset for the given percentatges. For each iteration a single trainset and dataset is build. As in the K -fold cross-validation the algorithm has been applied separately to each class to train each classifier.

This method does not ensure all instances are both used for training and detection, but in contrast to a full cross-validation procedure, random subsampling has been shown to be asymptotically consistent[9]. Predictions using this method give a realistic estimation of the classifier performance with external validation data, while K -fold cross-validation usually gives overly optimistic estimations[10].

Because of the diversity of the datasets that already offers random sub-sampling and its realistic estimation, the repeated random sub-sampling algorithm has been chosen for the experiment dataset builder.

7. Evaluation

Some of the input parameters of the trainer and the detector cannot have to be set, so a performance evaluation of the semantic shot classification models need to be performed first.

This chapter includes the different measures that have been developed in order to evaluate the classifier performance, the method used for evaluating the classifiers and the obtained results.

7.1. Developed measures

This section covers the metrics that have been developed in order to use the already explained measures for evaluating binary classifiers to also evaluate multiclass classifiers.

There have been developed seven functions in order to allow computing multiclass evaluations out of a confusion matrix. Table 7.1 lists the implemented methods, their input and output parameters and a general description. The explanation of each measure and how is it computed is described in the following sections.

Method	Input	Output	Description
measurePrecision	int classID	double	Computes the recall of a class
measureRecall	int classID	double	Computes the precision of a class
measureFBeta	int classID double beta	double	Computes the FB measure of a class
measureMPrecision	-	double	Computes the mean recall
measureMRecall	-	double	Computes the mean precision
measureMFBeta	double beta	double	Computes the mean FB measure
plus	ConfusionMatrix c2	ConfusionMatrix	Computes the summation of two confusion matrices

Table 7.1.: ConfusionMatrix methods

7.1.1. Precision and Recall out of a confusion matrix

The problem about evaluating a multiclass system in terms of precision and recall from a confusion matrix is that it only provides information about true and false positives instances, so precision can be measured but recall is unknown.

The solution adopted in order to get the true and false negatives has been rearranging the confusion matrix with an additional row and column for placing a negative class. This class would contain all the instances that do not belong to any of the classes and will provide the necessary information about the classifier performance when a shot type is not of interest.

Each “No class” cell, except for the “true negative” one, is both false positive of a class and false negative of the “No class”. This duality makes possible computing both precision and recall for all classes. When computing each class precision but the clutter class the false positive is used, while when computing recall the false negative is required. This can be seen on Table 7.2.

		Automatic			
		class1	class2	class3	No class
Manual	class1	<i>tp1</i>	<i>fp2</i>	<i>fp3</i>	<i>fp</i>
	class2	<i>fp1</i>	<i>tp2</i>	<i>fp3</i>	<i>fp</i>
	class3	<i>fp1</i>	<i>fp2</i>	<i>tp3</i>	<i>fp</i>
	No class	<i>fp1</i> <i>fn</i>	<i>fp2</i> <i>fn</i>	<i>fp3</i> <i>fn</i>	<i>tn</i>

Table 7.2.: Confusion matrix for multiclass classifiers

Now precision and recall can be measured for every class. Precision will be expressed as the true positive instances divided by the summatory of the class column instances while recall will be the true positive instances among all the instances contained in the class row.

$$Precision(i) = \frac{x(i, i)}{\sum_{n=1}^N x(i, n)} \quad (7.1)$$

$$Recall(i) = \frac{x(i, i)}{\sum_{n=1}^N x(n, i)} \quad (7.2)$$

Where:

- “i” corresponds to the current class among the total number of classes “N”.
- x(i, n) and x(n,i) corresponds to the value in that position of the confusion matrix.

The mean precision and mean recall can be computed by averaging the precision and recall results for each class:

$$MeanPrecision = \frac{1}{I} \sum_{i=1}^I \left(x(i, i) / \sum_{n=1}^N x(i, n) \right) \quad (7.3)$$

$$MeanRecall = \frac{1}{I} \sum_{i=1}^I \left(x(i, i) / \sum_{n=1}^N x(n, i) \right) \quad (7.4)$$

7.1.2. F1 and F β measure out of a confusion matrix

As precision and recall out of the confusion matrix has been resolved, this is the expression of the modified F1 and F β measure in terms of the precision and recall:

$$MF1(i) = 2 \cdot \frac{precision(i) \cdot recall(i)}{precision(i) + recall(i)} \quad (7.5)$$

$$MF_{\beta}(i) = (1 + \beta^2) \cdot \frac{precision(i) \cdot recall(i)}{(\beta^2 \cdot precision(i)) + recall(i)} \quad (7.6)$$

To obtain the mean F1 and mean F β measure, the average of all classes has to be computed:

$$meanMF1(i) = \frac{1}{I} \sum_{i=1}^I \left(2 \cdot \frac{precision(i) \cdot recall(i)}{precision(i) + recall(i)} \right) \quad (7.7)$$

7. Evaluation

$$meanMF_{\mathbb{k}}(i) = \frac{1}{I} \sum_{i=1}^I \left((1 + \mathbb{k}^2) \cdot \frac{precision(i) \cdot recall(i)}{(\mathbb{k}^2 \cdot precision(i)) + recall(i)} \right) \quad (7.8)$$

Because of the unified measure of precision and recall and the possibility to compute a unique value to describe the classifier's performance, the F1 measure has been the reference measure for evaluating the semantic shot classifier and obtaining the optimum results.

7.2. Evaluation methodology

The mentioned measures have been implemented to evaluate the semantic shot classifiers. The main problem has been finding the optimum values for the input parameters of each classifier .

The three parameters combinations has been computed within the specified ranges:

- **Minimum score:** from 0 to 1 in steps of 0.1.
- **Minimum number of elements:** from 1 to 5 in steps of 1.
- **Maximum radius:** from 0.1 to 1 in steps of 0.1.

For all possible combinations the minimum number of elements and the maximum radius are used as input parameters to generate the model to be used by the detector. When detecting, the minimum score value is used at the decision stage.

For each combination three iterations have been done using the repeated random sub-sampling method, so the classifier has been tested with three aleatory trainsets and test-sets for each combination. This is the measure that has been computed for each iteration:

- **Confusion matrix:** the confusion matrix has to be computed as all measures are based on this matrix analysis. On each iteration all detected instances are accumulated to the correspondent semantic shot with the already detected instances.

After all iterations, the following measures have been calculated out of the obtained confusion matrix:

- **Mean F1 measure:** this measure is the one used as a classifier performance comparator so it is calculated in order to determine the optimum combination.

- **F1 measure for each class:** each class is analyzed independently so it provides a performance estimation of each semantic shot.
- **Precision and recall for each class:** for each class precision and recall are also calculated to provide more accurate results on each semantic shot performance.

All results are stored in a .log file. After determining which is the parameters combination with the highest mean F1 measure, all measures related to the optimum combination are stated at the end of the file.

For each combination a bar graph showing the F1 measure, precision and recall of each semantic shot is also performed by modifying an already developed UPSeek Java class that uses JFreeChart¹, a free chart library, to automatically generate bar graphs and store them in a .PNG file.

Once all the process is done, a Matlab mesh plot is done using a matrix that stores all possible combinations of minimum number of elements and maximum radius within the optimum minimum score.

7.3. Results

This section shows the performance of both classifiers with the optimum values of the three classifier parameters: minimum score, minimum number of elements and maximum distance.

An analysis on how these parameters affect the classifier performance can be consulted on Annex A.

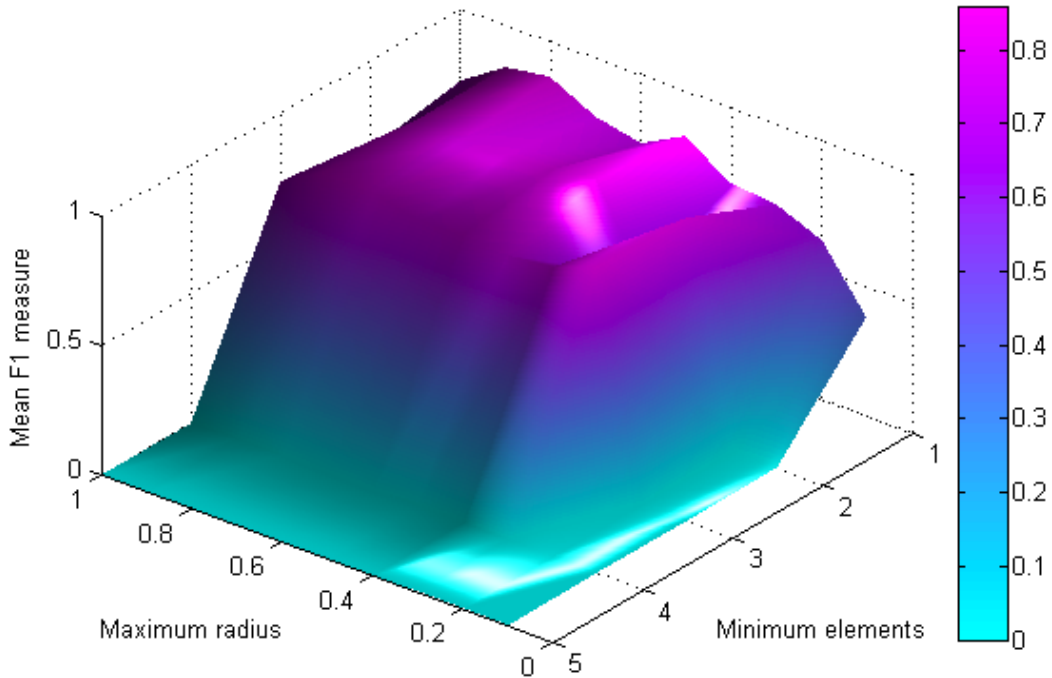
¹<http://www.jfree.org/jfreechart/>

7.3.1. Soccer matches optimum results

The maximum mean F1 measure is **0.8599863416562412** and corresponds to the following values of the three variables:

- Minimum score: 0.7
- Minimum number of elements: 1
- Maximum radius: 0.8

Figure 7.1 shows the mean F1 measure for all combinations computed of minimum elements and maximum distance with 0.7 minimum score (optimum).



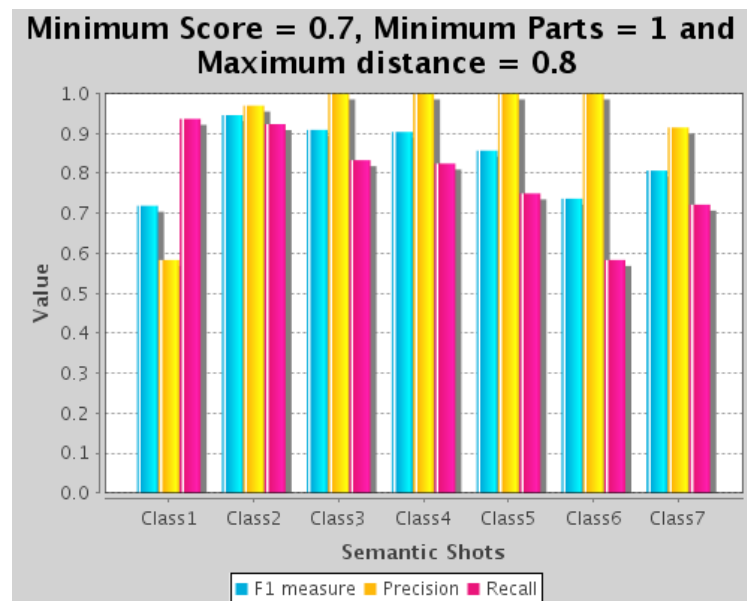
Min elements	Max radius									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
1	0.385	0.616	0.691	0.703	0.822	0.734	0.771	0.860	0.829	0.714
2	0	0	0.838	0.546	0.858	0.827	0.828	0.816	0.832	0.659
3	0	0	0.853	0.784	0.764	0.785	0.770	0.818	0.833	0.727
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0

Figure 7.1.: Soccer match F1 measures for 0.7 minimum score

The confusion matrix of the optimum combination can be seen on Table 7.3. From this confusion matrix the F1 measure, precision and recall for each class are extracted and represented on Figure 7.2.

		Automatic						
		Class1	Class2	Class3	Class4	Class5	Class6	Class7
Manual	Class1	652	11	0	0	0	0	33
	Class2	4	482	0	0	0	0	36
	Class3	1	0	20	0	0	0	3
	Class4	0	0	0	94	0	0	20
	Class5	0	0	0	0	18	0	6
	Class6	0	1	0	0	0	21	14
	Class7	461	3	0	0	0	0	1204

Table 7.3.: Best soccer match confusion matrix instances



	Class1	Class2	Class3	Class4	Class5	Class6	Class7
Precision	0.58	0.97	1.0	1.0	1.0	1.0	0.91
Recall	0.93	0.92	0.83	0.82	0.75	0.58	0.72
F1 measure	0.72	0.95	0.91	0.90	0.86	0.74	0.81

Figure 7.2.: Precision, recall and F1 measure bar graph for each soccer match class

7. Evaluation

When analyzing the confusion matrix it can be seen which classes have been detected as other classes. Figure 7.3 shows all classes which presents some detection problem, the class that causes that problem and the percentage error of the confusion.

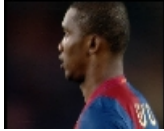









Manual	Automatic	% Error
Player close-up 	Player medium shot 	0.76%
Player medium shot 	Player close-up 	1.58%
Stadium overview 	Player medium shot 	4.16%
Box 	Player close-up 	2.77%
Clutter 	Player medium shot 	27.63%
Clutter 	Player close-up 	0.17%
All classes	Clutter	7.9%

Figure 7.3.: Soccer match detection errors

The highest percentage error (27.63%) corresponds to the clutter class being confused with the player medium shot. It can't be told the reason why this happens in advanced but it is probably related to the dominant color descriptor, as in many player medium shot labeled instances grass represents a high percentage of the shot.

A confusion which is more understandable is the one that affects the box class and the player close-up class. As it can be seen both show a close-up of a person and as no face recognition is applied the classifier cannot tell which one is a player or a VIP person.

The player close-up with the player medium shot confusion and viceversa are also quite understandable as they are really similar and sometimes it is hard to tell whether the shot size is a close up or a medium shot.

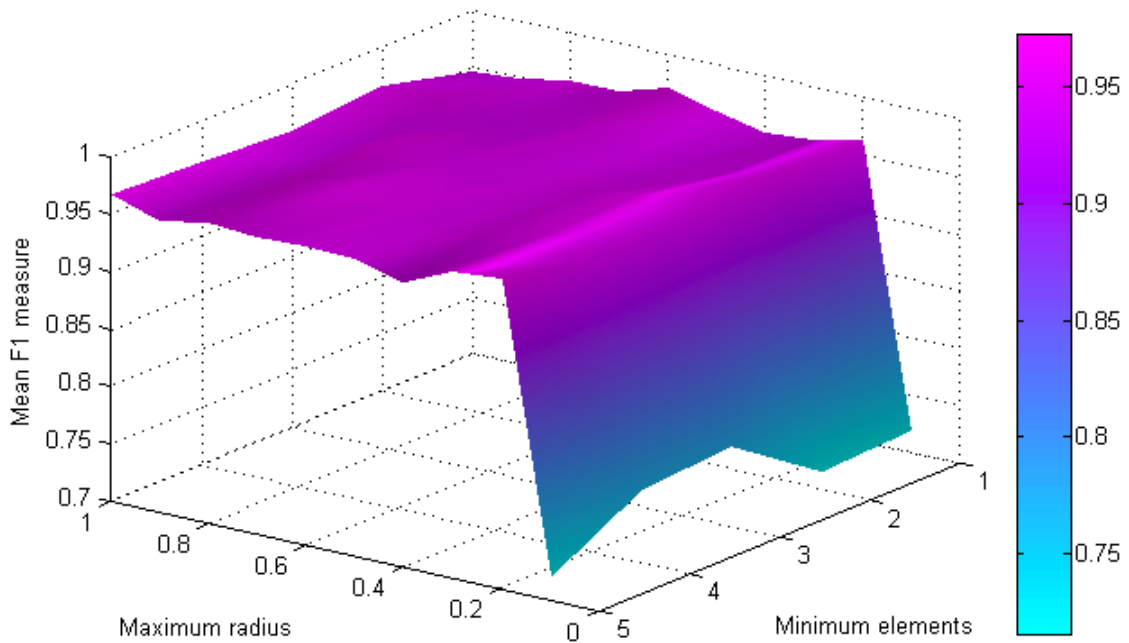
The stadium overview confusion has not a simple explanation, the most probable thing is that the trainer does not have enough positive instances to generate a proper model, as the stadium overview keyframes were really limited in number.

7.3.2. Catalan Parliament optimum results

The maximum mean F1 measure is **0.9720188262233014** and corresponds to the following values of the three variables:

- Minimum score: 0.9
- Minimum number of elements: 3
- Maximum radius: 0.8

Figure 7.4 shows the mean F1 measure for all combinations computed of minimum elements and maximum distance with 0.9 minimum score (optimum).



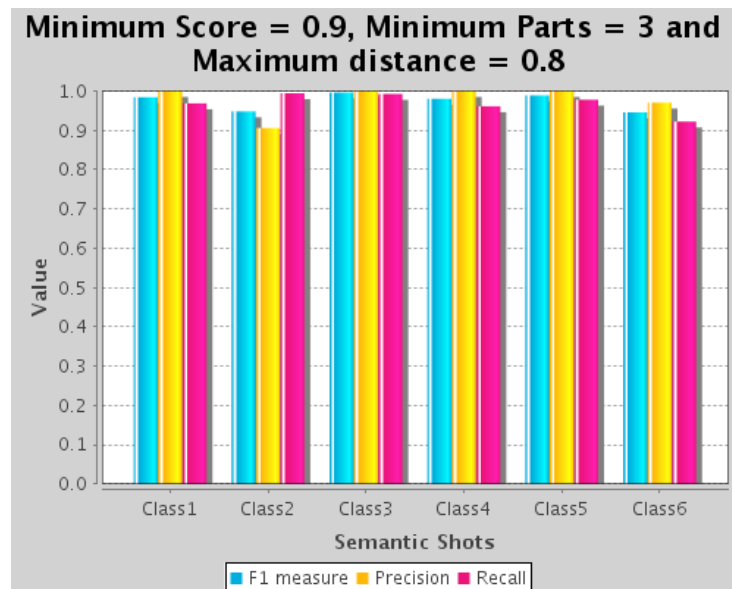
Min elements	Max radius									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
1	0.718	0.962	0.952	0.949	0.958	0.970	0.957	0.957	0.949	0.946
2	0.715	0.962	0.955	0.952	0.952	0.962	0.951	0.951	0.966	0.965
3	0.769	0.948	0.968	0.960	0.960	0.957	0.959	0.972	0.944	0.958
4	0.763	0.955	0.965	0.962	0.956	0.963	0.968	0.966	0.954	0.964
5	0.719	0.970	0.966	0.947	0.958	0.959	0.958	0.961	0.953	0.967

Figure 7.4.: Catalan Parliament F1 measures for 0.9 minimum score

The confusion matrix of the optimum combination can be seen on Table 7.4. From this confusion matrix the F1 measure, precision and recall for each class are extracted and represented on Figure 7.5.

		Automatic					
		Class1	Class2	Class3	Class4	Class5	Class6
Manual	Class1	93	0	0	0	0	3
	Class2	0	376	0	0	0	2
	Class3	0	0	405	0	0	3
	Class4	0	0	0	75	0	3
	Class5	0	0	0	0	135	3
	Class6	0	39	0	0	0	465

Table 7.4.: Best Catalan Parliament confusion matrix



	Class1	Class2	Class3	Class4	Class5	Class6
Precision	1.0	0.91	1.0	1.0	1.0	0.97
Recall	0.97	0.99	0.98	0.96	0.98	0.92
F1 measure	0.98	0.95	0.99	0.98	0.99	0.94

Figure 7.5.: Precision, recall and F1 measure bar graph for each Catalan Parliament class

When analyzing the confusion matrix it can be seen which classes have been detected as other classes. Figure 7.3 shows all classes which presents some detection problem, the class that causes that problem and the percentage error of the confusion.

7. Evaluation



Manual	Automatic	% Error
Clutter 	Close-up 	7.73%
All classes	Clutter	1.27%

Figure 7.6.: Catalan Parliament detection errors

The only confusions on detection are related to the close-up class as it is the semantic shot with more visual diversity as each close up shot is taken with different camera position and thus slightly different background.

Part II.

User Interface

8. State of the art

It has also been studied the state of the art of graphical interfaces that deal with annotation in terms of usability, specially those which include multiclass annotation and those providing tools to automatically suggest annotations so the user has only needs to validate them.

8.1. iPhoto

iPhoto is a software for Mac that allows viewing, organizing and editing photos. It has interesting tools such as face detection and recognition.

iPhoto automatically detects faces on photographs so that the user can add a name to each face. Once a name is introduced in the system it appears as a hint name when labeling faces. This can be seen on Figure 9.2.

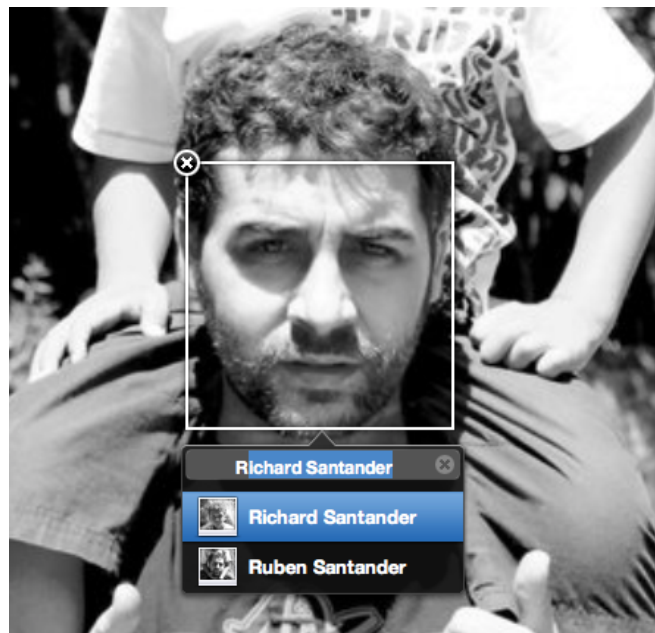


Figure 8.1.: iPhoto labeling

Once the user labels a person and click on his/her face in the “faces section”, iPhoto suggests lookalikes. Suggested faces have to be validated by clicking once to approve them (green label) or twice to reject them (red label).

Figure 9.3 shows an example of suggested faces validation where one of the suggested face doesn't match the person. On the top of the panel the hand-labeled faces are shown, while on the bottom of the panel are placed the suggested ones.

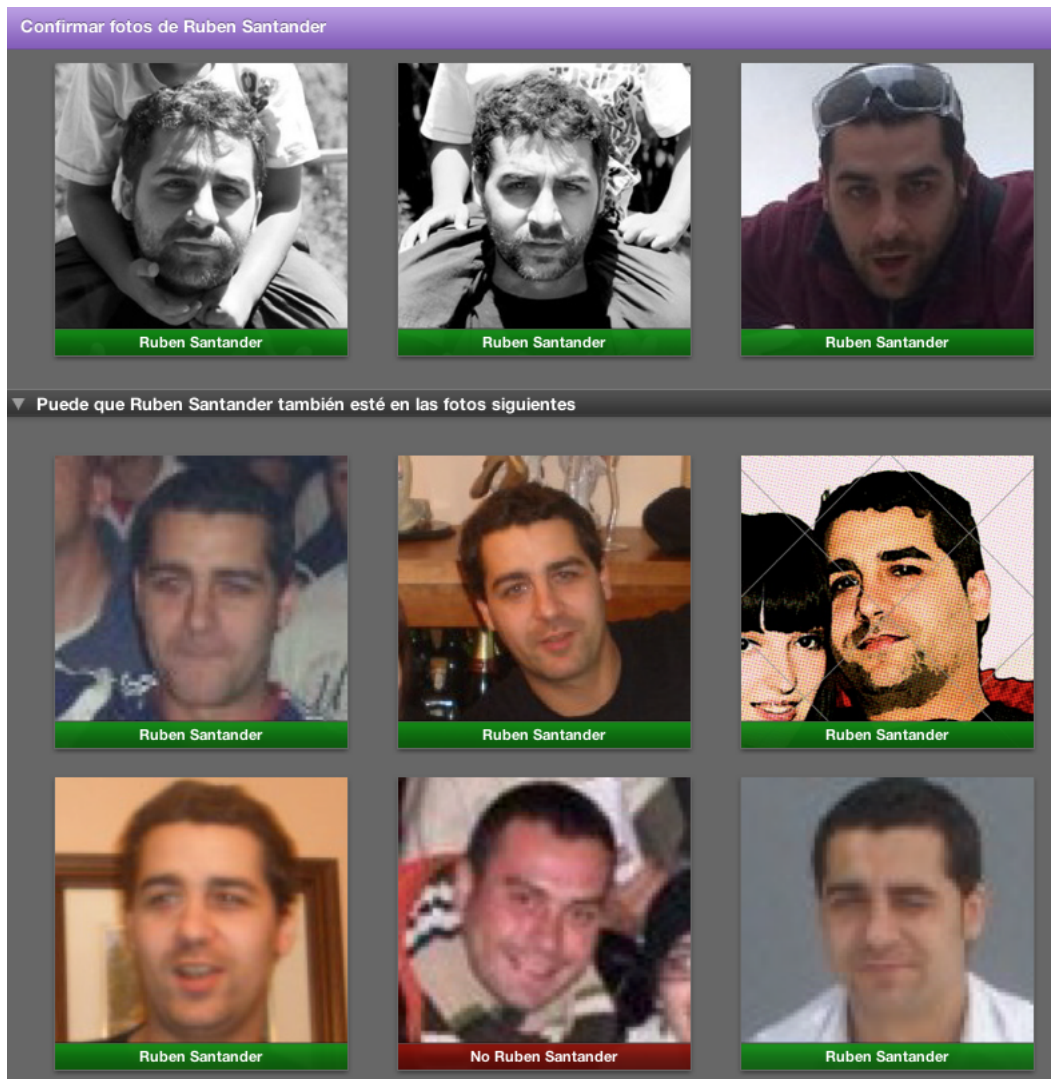


Figure 8.2.: iPhoto validation

After validation the system learns from these new labeled instances and suggests other photos with the new information about the person.

8.2. GAMERA

This interface is part of a Python-based toolkit for structured character recognition developed by the Johns Hopkins University[18].

It is intended to be used for training symbol recognition systems using supervised training. The main feature that speeds up the annotation task is the possibility of using the currently labeled data in the database to automatically classify the unlabeled data as in semi-supervised learning.

It also provides tools for symbol table creation, display of symbols lists, highlight symbols in context within a document and other features such as automatic selection of symbols for sequential classification that give flexibility to the interface, allowing the user to determine which is the most efficient way to work.

The interface not only provides an annotation tool but can also be used to modify the existing databases to change the classification model.

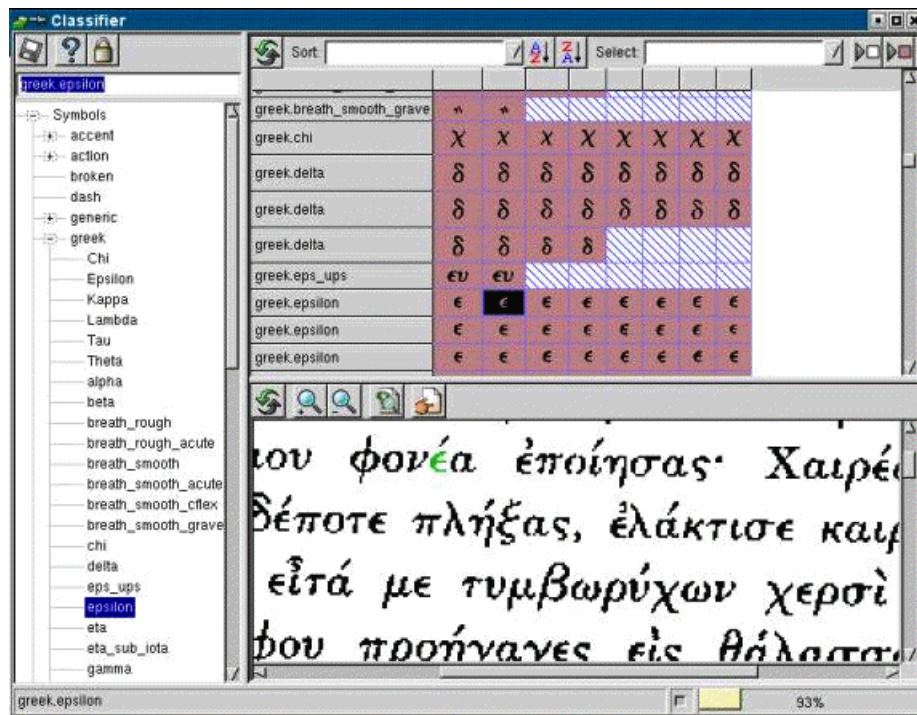


Figure 8.3.: Gamera interface screenshot for Greek characters annotation

8.3. Lookapp

Lookapp¹[19] is a system for the interactive construction of web-based concept detectors. It uses YouTube videos as training source and its detector is based on cloud computing platforms.

The application uses the user query to find similar tagged videos at YouTube² to use them as training sources. It uses Google App Engine³ for parallelized feature extraction and the Google Prediction API⁴ to construct the concept detector.

Figure 8.4 shows the lookapp retrieval analysis screen.

The screenshot shows the Lookapp interface for a retrieval analysis. At the top, there is a navigation bar with links for 'home', 'concepts', 'detectors', and 'about', and the Lookapp logo. Below this, the user's query is 'Sagrada Familia', and the concept is 'auge'. There are controls for 'Full Text', 'Category' (set to 'None'), 'Time' (set to 'all time'), and 'Unique User' (checked). A 'Retrieve Videos' button is visible.

The main section is titled 'Retrieval Analysis:' and includes links for 'retrieval list', 'relevance feedback', and 'print report'. There are also controls for 'descriptive name', 'Color Hist', 'Google Prediction API', and 'Build Detector'.

The interface is divided into two main columns:

- Keyframes:** A grid of 100 small image thumbnails showing various scenes related to the Sagrada Familia, including architectural details, people, and surrounding areas.
- Tags:** A list of tags generated by the system, including 'wgbh', 'gaudi', 'art', 'familia', 'family', 'statue', 'vlog', 'sagrada familia', 'catalunya', 'shape', 'cataluña', 'church', 'cathedral', 'catalunha', 'trip', 'familia', 'evangelists', 'pope', 'la travel', 'manning', 'blog', 'españa', 'surface', 'products', 'the world', 'paps', 'agbar', 'octopus', 'products', 'td', 'iberisch', 'john', 'stock', 'mercury', 'religion', 'park', 'holy park', 'photography', 'fiesta', 'alan', 'santa', 'gaudi', 'interior', 'details', 'basilica', 'sagrada', 'guide', 'croisiere', 'tourism', 'abis', 'spain', 'catalonia', 'barcelona', 'sagrada', 'casa', 'mila', 'balconies', 'catalano', 'catalonian', 'catalan', 'igreja', 'antoni', 'sagrada familia', 'info', 'parsons', 'barcelona', 'antonio', 'project', 'красы', 'miles', 'architecture', 'mila', 'antoni', 'gaudi', 'クルーズ', 'kreuzfahrt'.

Below the tags, there are two green boxes:

- Suggested Keywords:** familia barcelona spain madrid de la architecture españa gaudi ibiza arquitectura seville el catalunya en
- Synonyms:** (This box is currently empty).

On the right side, there is a **Categories** section with a pie chart showing the distribution of video categories. The categories include: Music, News / Politics, Education, Film / Animation, People / Blogs, Travel / Events, Howto / Style, Entertainment, Movies, Sports, and Movies / Style. The pie chart shows that 'Travel / Events' and 'Music' are the most prominent categories.

At the bottom, there is a **Suggested Category:** section with a green box containing the text 'None'.

Figure 8.4.: Retrieval analysis of the query “Sagrada Familia” screenshot

¹<http://lookapp.appspot.com>

²<http://www.youtube.com/>

³<http://code.google.com/intl/en/appengine/>

⁴<http://code.google.com/apis/predict/>

8. State of the art

On the left of the retrieval analysis view there is mosaic that displays the retrieved keyframes for the current query. It also provides tag and category suggestions at the right side of the interface so the user can find the concept he or she is looking for.

The “relevance feedback” view allows labeling the retrieved sets as positive or negative instances in order to train the classifier. This labeling is done by selecting the positive keyframe sets among the negative ones. Figure 8.5 shows a screenshot of a relevance feedback example.

The screenshot shows a web interface titled "Relevance Feedback:*" with two tabs: "retrieval list" and "retrieval analysis". The "retrieval analysis" tab is active. At the top right, there are three buttons: "descriptive name", "Color Hist", and "Google Prediction API", followed by a "Build Detector" button. Below these, the "Positive" tags are "barcelona", "gaudi", "familia", and "sagrada", and the "Negative" tags are "project", "john miles", and "parsons alan". A "Suggested Query" box contains "Sagrada Familia barcelona -project -parsons -alan".

The main area displays a grid of keyframes. A message at the top says "Select keyframes by clicking. Hold 'ctrl' to select multiple." and "You've selected 18 out of 47 keyframes". A "Provide Feedback" button is located at the top right of the grid. The keyframes are arranged in a grid, with some highlighted in green (selected) and others in red (not selected). The keyframes show various scenes of the Sagrada Familia, including the exterior, interior, and people.

Figure 8.5.: “Sagrada Familia” retrieval analysis screenshot

8.4. IM3I

The IM3I[20] is an adaptable multimedia repository software that derives semantic descriptions from media assets. It was developed in 2008-2010 under the FP7 programme⁵.

IM3I offers a set of web services and rich internet applications developed in Adobe Flex that allows:

- Automatic audio and video annotation.
- Semantic semi-automatic annotation based on ontologies
- Semantic searches.
- Authoring and publishing.

Figure 8.6 shows the web-based tool provided to validate automatic annotations, which can be removed by clicking on the negative icon of each tag. Other tags for which no concept detector has been trained yet can be introduced by clicking the positive icon and writing the new concept in the text box.

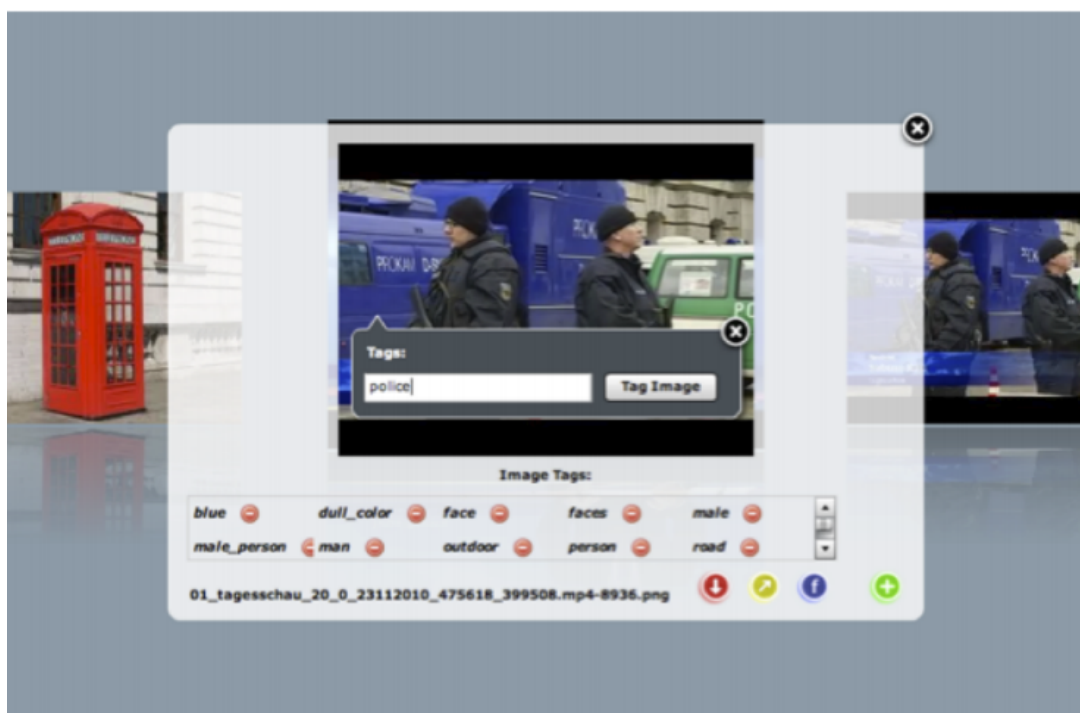


Figure 8.6.: IM3I semi-automatic-annotations screenshot

⁵The FP7(7th Framework Programme for Research and Technological Development) is a European Commission programme that offers research funding

8.5. GAT

GAT⁶ is a manual annotation tool of still images at region and graphic level developed in Java by the Image Processing Group at the UPC[14].

GAT provides a interface to create new binary or multiclass ontologies. It also provides a graphic annotation interface of keyframes to create instances of the semantic class to generate the semantic model. Instances can be positive, negative or neutral:

- **Positive** (green): keyframes are labeled as positive if the semantic concept in the ontology does appear in the image.
- **Negative** (red): keyframes are labeled as negative if the semantic concept does not appear.
- **Neutral** (yellow): keyframes are labeled as neutral when the image is neither positive or negative, as the concept may appear but not in the expected way (for example an image containing the concept fading into another that does not contain it).

To ease the annotation process, beyond labeling individual keyframes, it provides tools that allow selecting all keyframes either as positive, negative or neutral and multiple selection from one keyframe position to another. Validation is done by a clicking the right mouse button.

Beyond generating semantic models, GAT is able to perform binary classifiers evaluation using cross-validation and shows the true/false positive/negatives instances, the precision and the recall of the classifier.

Figure 8.7 shows a screenshot of a multiclass ontology after annotation where the current class has 22 positive instances.

⁶<http://upseek.upc.edu/gat/>

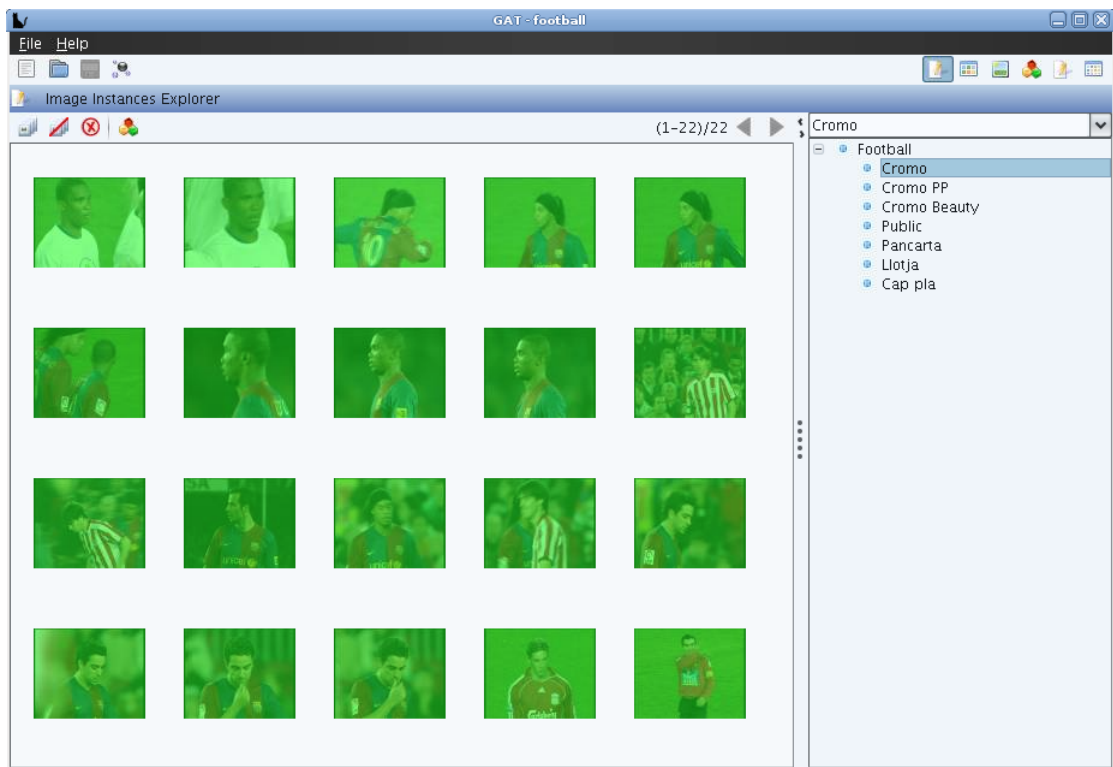


Figure 8.7.: Screenshot showing the positive instances of a semantic class

9. Design

This chapter covers all the design information related to the GUI developed at the CCMA, from the features conception to its work flow and the layout needs.

9.1. Functionalities and layout needs

The GUI can be divided in two basic functionalities: semi-automatic annotation and correction/validation. These functionalities are chronological and define the interface work flow:

1. **Semi-automatic annotation:** each keyframe semantic shot is detected by the classifier. A graphic way of showing the detections is needed.
2. **Fix errors and validate:** if the proposed semantic shots do not match the actual ones, the user will have to interact with the GUI and fix it. When errors are fixed then validation for the rest of the keyframes is required. A set of tools need to be defined for this task.

The next subsections describe the designs proposed for each one of the mentioned functionalities and the solution chosen for the development stage.

9.1.1. Semi-automatic detections

Two different layout proposals for showing the keyframes detected semantic shot were presented to the Documentation Department at the CCMA in order to keep developing the interface with the structure that better suit their needs.

The first proposal (Figure 9.1) was based on showing all keyframes with the same semantic shot in the same tab, having as many tabs as semantic shots. Direct consequences of working with this design are:

- The number of keyframes on each tab is fixed but the number of pages for each tab is variable as it depends on the quantity of keyframes detected for that shot-type.
- Keyframes can be labeled as positive (detection is ok) or negative (detection is wrong) .
- Besides labeling as negative, the information of the correct shot-type needs to be provided.
- Only one shot-type can be seen a time.

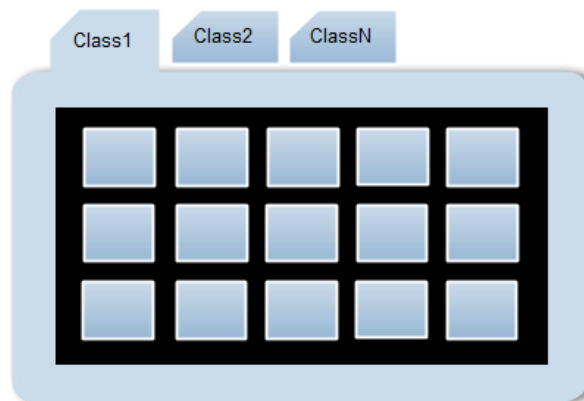


Figure 9.1.: First layout proposal

The second proposal (Figure 9.2), thought to be more effective when correcting errors, included in the same tab as many keyframe rows as semantic shots has the current domain. This made possible having in the same page all semantic shots classified.

These are some of its advantages and disadvantages:

- The columns of each row are fixed for all shot types. As not all shot types have the same keyframes detected and pagination is shared by all shot-types, some of the rows can run out of keyframes.
- This layout makes it suitable for drag and drop functionalities to easily correct a keyframe shot-type detection, so there is no need of labeling any keyframe as negative, just drop it to the right place.
- Once all keyframes are in the right place the page can be entirely validated.

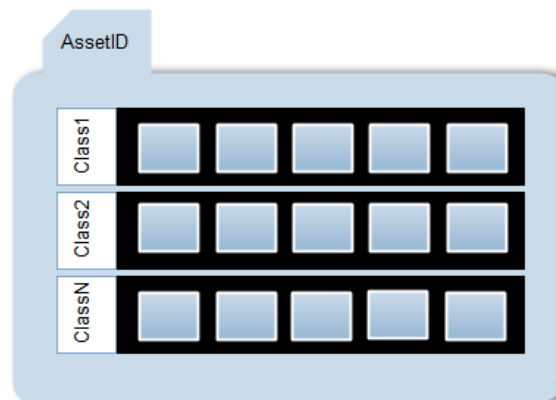


Figure 9.2.: Second layout proposal

As this layout offered the possibility of working with functionalities such as drag and drop, the second proposal was chosen by the CCMA.

9.1.2. Fix errors and validate

The **drag and drop** has been chosen to be the primary tool for fixing wrong detections. These are all the possible drag and drop uses that have been thought for the interface to be helpful:

- Drag a keyframe from a panel and drop it to another
- Drag multiple keyframes from the same panel and drop them to another using the control key
- Drag multiple keyframes from several panels and drop them to a another panel using the control key

Two different methods have been implemented to validate detections and both of them have been implemented to give the user more flexibility:

- **Validate page:** this choice would validate the entire visible page.
- **Validate semantic shot:** this choice would only validate a specific semantic shot

After validation the Documentation Department at the CCMA was interested in having the possibility of seeing all validated keyframes organized by semantic shots and being able to mark some of them as top keyframes of interest. To do so additional tabs were designed to show all validated detections.

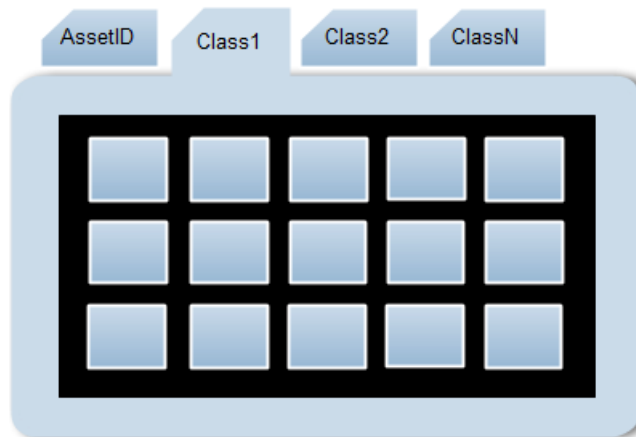


Figure 9.3.: Definitive layout semantic shot view

Another tools have been design to help rearranging the keyframes as needed:

- **Sort by score:** it allows to see detections sorted by score from higher to lower in each semantic shot panel, so the first pages would ideally contain less errors and thus would be easier to correct and validate.
- **Set minimum score:** setting a minimum score would make all keyframes with the detection score under it, be placed in the clutter class panel (No class). This tool can be used after sorting by score so it can be seen the minimum score at which the classifier is working properly.

This tools need to be organized and allow the user to interact with them by configuring parameters or raising events.

9.2. Work flow

When running the application at first, only the menu is shown and partially disabled so the user is forced to choose a domain and then leads him or her to enter an asset ID by activating and focusing the asset text box.

After an asset is chosen it must be checked wether the asset has already been ingested into the database or not. If that asset is avabile in the database a preview of its keyframes will be shown while if it is not avabile an error message would alert that the asset ID introduced is not a valid one. After the preview the user can proceed to annotate by clicking the “Accept” button choose another asset by clicking the “Cancel” one.

9. Design

When accepting to annotate the keyframes on the preview, semi-automatic annotation is done. At this point, the user can use all tools provided to help fixing detection errors in the order they want and as many times as it is needed.

After detecting, validation options are activated so the user can save the changes that has made and this information can be used to retrain the system and to save annotations.

The explained GUI work flow can be seen on Figure 9.4.

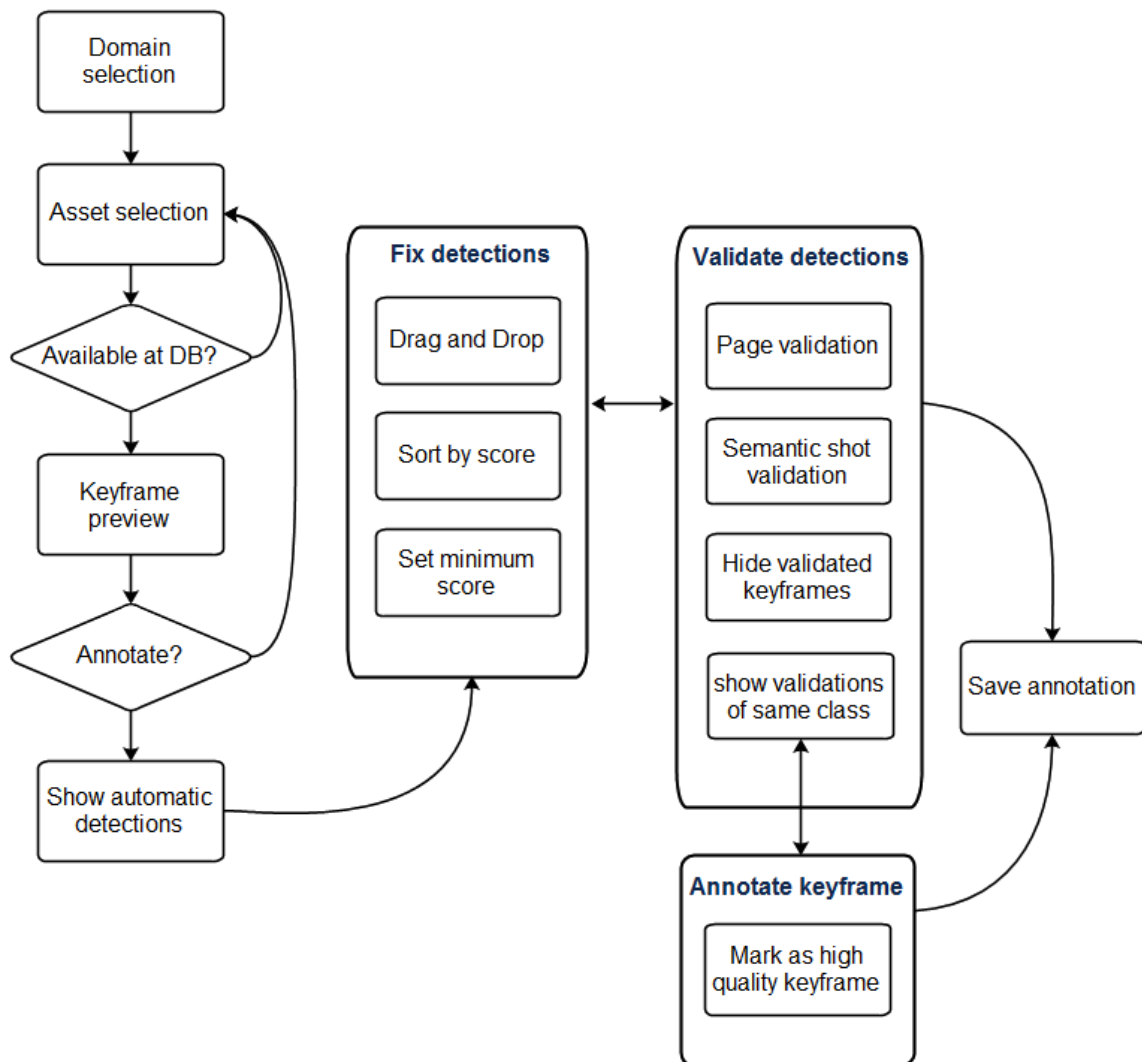


Figure 9.4.: Interface work flow diagram

10. Development

10.1. Development environment

This chapter covers the primary tools needed for developing the GUI and the languages that have been used.

10.1.1. GWT 2.2.0 - Eclipse Helios

GWT SDK allows programming in Java an AJAX front-end and then cross-compiles it into optimized JavaScript that automatically works across all major browsers.

Google provides a plugin for Eclipse that can be installed by using the following update site: <http://dl.google.com/eclipse/plugin/3.6>.

GWT allows working in two different modes:

- **Development mode:** this mode creates a Web Application launch configuration and launches it. The web application launch configuration starts a local web server and GWT development mode server. GWT provides the URL for the development mode server as it can be seen on Figure 10.1. Then the URL has to be pasted into a browser with the GWT Developer Plugin already installed as it is shown in Figure 10.2.

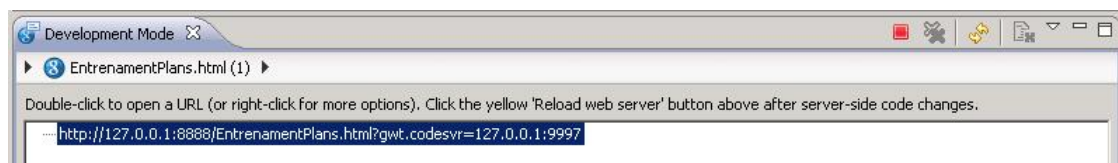


Figure 10.1.: Eclipse URL generation for development mode



Figure 10.2.: Google Chrome developer mode

- **Production mode:** to run the application as JavaScript and HTML the GWT the application needs to be compiled. The GWT compiler generates a number of JavaScript and HTML files from the MyWebApp Java source code in the MyWebApp/war/ subdirectory. To see the final application the MyWebApp/war/MyWebApp.html file has to be opened in a web browser, with no need of having any plug-ins.

the *development mode* has been used for the development of the application as it allows making changes just with saving the Java code without compiling it. On the other hand, production mode has been necessary to check the real speed of the application as some features such as drag and drop have a heavy delay in development mode and are optimized to work in production mode.

The main languages used in the Eclipse project apart from Java GWT have been the following ones:

- **JavaScript:** it has been used to create the overlay types to deal with JSON messages. This part is explained in the next chapter.
- **CSS:** it has been used to define the layout and appearance of the application.
- **XML:** it has been used to configure and map the proxy explained in the next chapter.

10.1.2. Google Chrome

For developing the GUI the Google Chrome browser has been chosen as it is the browser that the company recommended me to use as it is also from Google. However, other browsers have been tested too in order to ensure the application is working in all of them.

The Google Web Toolkit development plug-in is needed to run the application in *development mode*. Connections can be configured through this plug-in to allow cross-machine debugging, though it only connects to local machine by default.

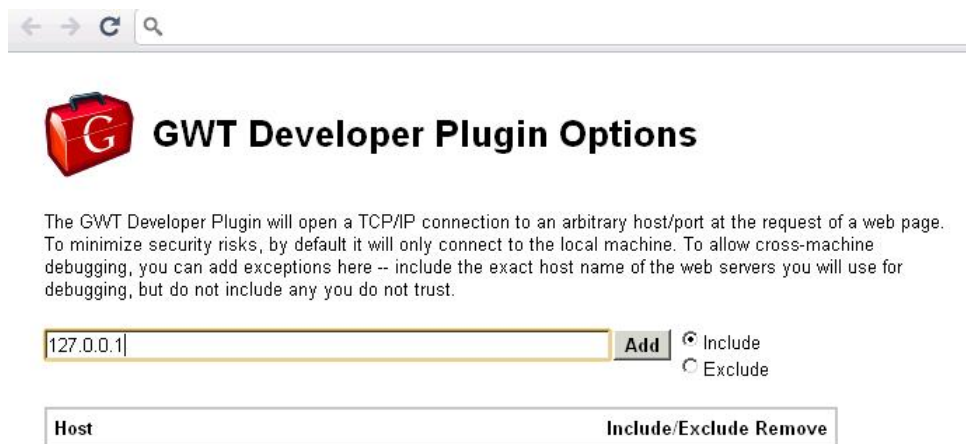


Figure 10.3.: Google Chrome developer plug-in

10.2. Communications

10.2.1. HTTP

HTTP (Hypertext Transfer Protocol) is a networking protocol for distributed, collaborative, hypermedia information systems. HTTP functions as a request-response protocol in the client-server computing model. The client submits an HTTP request message to the server. The server, which provides resources, returns a response message to the client. A response contains completion status information about the request and may contain any content requested by the client in its message body. In our case this content is a JSON message.

HTTP has different request methods, indicating the desired action to be performed on the identified resource. For this project two different methods have been used:

- **GET:** requests a representation of the specified resource, used for retrieval information.
- **POST:** submits the data to be processed to the identified resource. The data is included in the body of the request.

A typical URL has the following structure:

```
http://server:port/path/program?parameter1=value1&parameter2=value2
```

A query string is the part of a URL that contains data to be passed to web applications (all parameters and their value).

10.2.2. JSON

JSON (JavaScript Object Notation) is a text-based open standard designed for human-readable data interchange. It is derived from the JavaScript scripting language for representing simple data structures and associative arrays, called objects.

JSON is built on two structures:

- A collection of name/value pairs which are position-independent.
- An ordered list of values.

10.2.3. PROXY

The UPC provides a set of web services to be used by this GWT application that works as the client. An HTTP request from the client is submitted to get access to the web service resources. The response from the server is received in a JSON format. To avoid cross-domain problems related to security that causes using JSON from another domain, the CCMA has configured a proxy in the client to make the call in the local application and then redirecting it to the UPC through the main company's proxy, getting the response in the same application without any trouble.

As the call has to be done from the application client the server and port of the call has to be the server and port where the application is running. Both of them can be get automatically using these GWT functions:

```
//Get automatically the host and port of the client
String hostClient = Window.Location.getHost();
String portClient = Window.Location.getPort();
```

After getting both host and port of the client, a redirection needs to be done using a servlet class that extends HttpServlet class and that includes in the path the server, port and path at the UPC.

The call then has the following structure:

```
http://hostClient:portClient/remoteServer/remotePath/program?parameter1=value1&parameter2=value2
```

Where the remoteServer is the server that redirects the call at the CCMA and the remotePath is the path to the UPC where the webservices are hosted: `upseek.upc.edu:8080/upseek/`

10.2.4. Web services

Three different web services are used. First the user introduces an asset and a call is made in order to get all keyframes for that asset. Then a single call is made for every keyframe in order to obtain their class. After keyframes classes are validated or corrected the user can call the third service in order to retrain the classifier. The next scheme shows each web service structure.

Get data stream

For a given asset returns the title and url of the thumbnails of all keyframes belonging to that class.

- **Direct call:** `http://127.0.0.8888/upc.edu:8080/upseek/getdatastream?source=1008`
- **Redirected call:** `http://127.0.0.8888/cache.local/upc.edu:8080/upseek/getdatastream?source=1008`
- **HTTP method:** GET
- **Response:**

```
{ "results": [
  { "keyframe": "00_14_27_06",
    "URL": "http://147.83.50.77:8080/.../i3mam:25165/thumbnail"
  },
  { "keyframe": "00_09_44_20",
    "URL": "http://147.83.50.77:8080/.../i3mam:25167/thumbnail"
  }
  ...
],
  "success": "success"
}
```

Detection

For a given keyframe and ontology returns the class obtained in the detection.

10. Development

- **Direct call:** `http://upseek.upc.edu:8080/upseek/detection?source=1319&title=00_00_25_16&ontology=football`
- **Redirected call:** `http://127.0.0.1:8888/cache.local/upc.edu:8080/upseek/detection?source=1319&title=00_00_25_16&ontology=football`
- **HTTP method:** GET
- **Response:**

```
{ "success":true,
  "errorMessage":null,
  "ontologyStr":null,
  "detections":[
    { "cluster":0,
      "source":null,
      "title":null,
      "score":0.9886459708213806,
      "mask":null,
      "classId":2,
      "annotationLabel":false
    }
  ],
  "partsMaxDistance":0.0,
  "partsMinAmount":0
}
```

Train

For a given JSON object with ClassifierSO.java structure containing all validated keyframes and information about them, the server is able to retrain the system. The object that the application sends to the server has the following attributes:

```
public boolean success;
public String errorMessage;
private Vector<RetornUpseek> detections;
private String ontologyStr;
private double partsMaxDistance;
private int partsMinAmount;
```

- **Direct call:** `http://upseek.upc.edu:8080/upseek/train`

- **Redirected call:** `http://127.0.0:8888/cache.local/upc.edu:8080/upseek/upseek/train`
- **HTTP method:** POST

10.2.5. Communications in GWT

When calling a web service a `RequestBuilder` instance has to be created in order to specify the HTTP method (GET or POST for this project) and the URL of the web service.

Here there is an example of a GET HTTP call with code omitted for clarity.

```
final String url = "http://hostClient:portClient/remoteServer/
remotePath/program?parameter1=value1&parameter2=value2";
RequestBuilder req = new RequestBuilder(RequestBuilder.GET, UrlWebService));
try{

    Request request = req.sendRequest(null, new RequestCallback(){
        @Override

        public void onResponseReceived(Request request, Response response) {
            if (200 == response.getStatusCode()) {
                //Response OK
            } else {
                //Response error
            }
        }
        @Override
        public void onError(Request request, Throwable exception) {
            //No response
        }
    });
} catch (RequestException e) {
    e.printStackTrace();
}
```

On response received it has to be checked that the HTTP Status is 200 as this would mean the request has succeeded, while getting other status code would mean there has been any problem.

When the response is not even received due a limit of time exceeded or a connexion problem an exception is thrown.

Overlay types

The JSON string has to be transformed into a JavaScript object to work with JSON in GWT . Once the information is converted to a JavaScript object, there is free access to any of the message fields through the eval() function already implemented in JavaScript.

If information does not only need to be read but manipulated, GWT allows using “overlay types”, a subclass of a JavaScriptObject, to transform these JavaScript objects into Java objects. The following lines of code are extracted of the overlay type developed for the detection web service response.

```
public class DetectionResponse extends JavaScriptObject {

    protected DetectionResponse() {
    }

    //Getters
    public final native String getOntologyStr()/*- {
        return this.ontologyStr;
    }-*/;
    public final native JsArray<Entry> getEntries()/*- {
        return this.detections;
    }-*/;

    //Setters
    public final native void setOntologyStr(String ontologyStr)/*- {
        this.ontologyStr = ontologyStr;
    }-*/;
    public final native void setEntries(JsArray<Entry> detections)/*- {
        this.detections = detections;
    }-*/;
}
```

10.3. Data structure

10.3.1. Keyframe class

The Keyframe is the basic object of the data structure of this project. It includes three essential information layers:

- Information related to the Image, as the URL, the time code, the asset number where it belongs, etc.
- Semantic shot information got at the detection stage: score, semantic class.
- Validation information like if it has been validated or annotated as a quality keyframe.

Table 10.1 lists all keyframe attributes. The methods implemented in the keyframe class corresponds to the getter and setter for each attribute.

Variable	Type	Description
thumbnail	Image	Keyframe URL image
asset	String	AssetId
title	String	Keyframe time code
score	double	Detection score
classId	int	Detected class
isValidated	boolean	Tells if it has been validated
isAnnotated	boolean	Tells if it has been annotated as a quality keyframe

Table 10.1.: Keyframe class variables

10.3.2. KeyframeVectors class

The KeyframeVector class organizes all keyframes detections so all changes from the proposed semantic shot to the right one can be tracked. Every time an action that affects the keyframe detections is done by the user, the KeyframeVectors have to be reorganized. This structure is similar to the visual one, where all keyframes are arranged in a semantic shot row depending on its class. This makes it easier to apply all visual changes to the data structure.

KeyframeVectors attributes and defined methods can be seen on table 10.2 and 10.3.

Variable	Type	Description
numShots	int	Total number of semantic shots
keyframeVectorArray	ArrayList<Vector<Keyframe>>	Arraylist with a vector of keyframes for each semantic shot
trainingVectors	ArrayList<Vector<Keyframe>>	Arraylist with a vector of validated keyframes for each semantic shot

Table 10.2.: KeyframeVectors class variables

Method	Input Parameters	Type	Output
Constructor	numPlans	int	void
sortKeyframes	-	-	void

Table 10.3.: KeyframeVectors Methods

The KeyframeVectors constructor only needs a number of semantic shots to be entered and generates the keyframeVectorArray with as many Keyframe vectors as number of semantic shots.

The .sortKeyframes() method sorts each vector of the keyframeVectorArray independently.

10.4. Features development

This section explains all features development, mentioning the widgets that have been used and its visual result. In some of the features interesting implementations that have been important for the development of the GUI are also explained in detail.

10.4.1. Domain selection

To allow the domain selection a dropBox widget has been created with a “select domain” message on it. When the box is dropped the two available domains appear in the list

and one of them has to be selected. Figure 10.4 shows the drop box while choosing the Parliament domain.



Figure 10.4.: Domain drop box

10.4.2. Search by asset

To allow the user selecting the asset that want to annotate a text box widget has been created so the assetID can be written through the keyboard. Figure 10.5 shows the asset menu where the asset ID 1749 is being introduced.

A custom button has been created in order to introduce that asset number to the system and call the *get data stream* web-service to see if that asset is available and if so get all its keyframes. An alternative to do so has been provided by pressing the Intro key while in the text box.

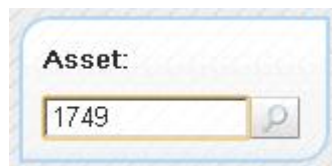


Figure 10.5.: AssetID text box and search button

10.4.3. Asset preview

The preview panel has been made by creating a pop-up panel with a scroll panel in it and a grid showing all keyframes. It also includes the “Accept” and “Cancel” buttons in order to annotate or choose another asset respectively.



Figure 10.6.: Keyframes preview

10.4.4. Error message

When the requested asset is not ingested in the database the web service response indicates so with an error message. The error message is shown in a pop-up panel like the one in Figure 10.7.

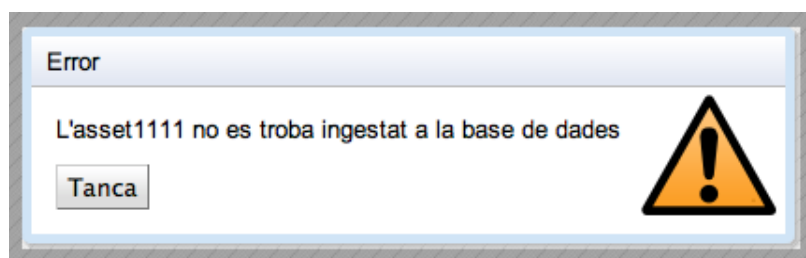


Figure 10.7.: Error message: the requested asset is not ingested in the database

The message has to be closed with the provided “Close” button so the user can introduce a different asset ID.

10.4.5. Drag and drop

Two different areas and drag and drop controllers have been created to allow drag and drop. There is a single boundary drag area that includes all the panels, as multiple drag in different panels is allowed.

For each boundary panel a drag controller has to be set so draggable widgets can be picked up and moved around. However, as drop is only allowed at one of the horizontal panels at a time, each drop target within the boundary panel is associated to a drop controller, so there would be as many drop controllers as semantic classes (including the clutter class).

Drag and drop attributes can be seen on Table 10.4.

Variable	Type	Description
boundaryPanel	AbsolutePanel	Draggable area
targetPanels	Vector<HorizontalPanel>	Vector including all Droppable areas
dragController	PickupDragController	Controller to pickup&move keyframes across the boundary panel
dropControllers	Vector<DropController>	Controllers to drop keyframes into the target panels areas
classIdDrag	Vector<int>	From where is/are the keyframe/s dragged
classIdDrop	int	Where is/are the keyframe/s dropped
draggedKeyframes	Vector<Keyframe>	Vector of multiple dragged keyframes

Table 10.4.: DragAndDrop class variables

Two methods have been developed to ease the use of drag and drop features: the constructor and a makeDraggable method.

The DragAndDrop constructor has been implemented to automatically generate a unique dragController and a dropController for each panel. Then it registers all the panels to that dragController, allowing dragging any draggable widget inside the boundaryPanel to any of the panels.

Having one dragController for each page caused problems at dragging (visual drag and drop was allowed but the handlers attached to the dragController were not), so Horizontal panels are always the same, and keyframes have to be added and removed from them, allowing to create a unique dragController for all pages as needed.

Besides the setter and the getter methods for each DragAndDrop variable, it has been set the makeDraggable method, which gets the keyframe thumbnail and makes it draggable from the boundaryPanel and droppable to any of the panels.

Method	Input Parameters	Type	Output
Constructor	boundaryPanel targetPanels	AbsolutePanel Vector<HorizontalPanel>	void
makeDraggable	keyframe	Keyframe	void

Table 10.5.: DragAndDrop Methods

Two handlers have been set to rearrange the keyframes on the panel, keeping the same number of keyframes for each panel.

On drag start event:

- For each panel where a drag action starts as many keyframes as have been dragged are added in order to fill the blanks with the keyframes in the next page (if there is next page).
- The drag and drop draggedKeyframes vector is filled with the selected keyframes.
- The drag and drop classIdDrag vector is also filled with the classID of the dragged-Keyframes.
- The draggedKeyframes are erased from the panel with its classIdDrag.

On drag end event:

- When a drag action ends if the panel is full as many keyframes as have been dropped have to be moved to the next page in order to let the dropped keyframes fit in the panel.
- The drag and drop draggedKeyframes vector is filled with the selected keyframes.
- The drag and drop classIdDrop is refreshed with the classID of the panel where keyframes have been dropped.
- The draggedKeyframes are added to the classIdDrop panel and the keyframe classID attribute is changed for the classIdDrop number.

- Set all draggedKeyframes score to 100% and apply the validation CSS style, as the keyframes are now validated.
- Add draggedKeyframes to the correspondent classIdDrop vector position of the trainingVectors.

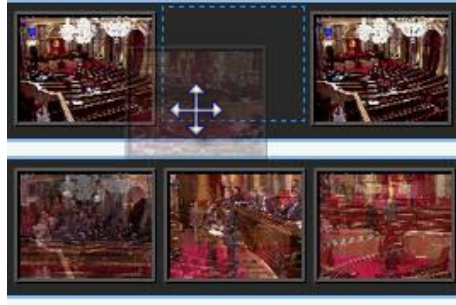


Figure 10.8.: Keyframe dragging

10.4.6. Set the minimum score

The interface allows the user to set a minimum score value from 0% to 100% in 10% steps for all detections as it can be seen in Figure 10.9. If the score of all semantic shot detectors is under the minimum score then the class assigned would be the clutter class. The default value of the minimum score is the experimentally found optimum value.

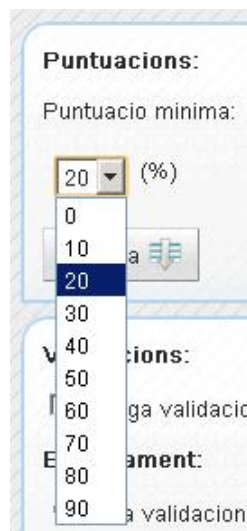


Figure 10.9.: Minimum score drop box

In order to allow users changing the minimum score once detections have been done, the minimum score at the detector has to be set to 0.0, as it is important to know which class

10. Development

has been detected with the lowest score (0.0) so the user can set it to all values, otherwise it would be impossible to know the detected class for a new minimum score lower than the value set on the detection step.

After each detection, a keyframe is assigned to the detected class or the clutter class depending on the minimum score value, keeping the information of the class and score obtained at detection to allow new minimum score values. When a user changes this value, then keyframes are rearranged depending on the new minimum score value.

10.4.7. Sort by score

To sort by score only a button is needed as there are no parameters to be given to the application.

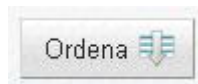


Figure 10.10.: Sort button

The following figures show a fragment of close-up Parliament keyframes before and after sorting them.



Figure 10.11.: Unsorted close-up semantic shot



Figure 10.12.: Sorted close-up semantic shot

Figure 10.11 does not show any close-up as the order of the keyframes has nothing to do with their score. Figure 10.12 contains the close-up keyframes with the highest score and detections are right.

The process for sorting all keyframes by score is done separately for each semantic shot vector. To be able to compare keyframes between them in terms of score and use the method `.sort()` that the `Vector` class provides, a `Keyframe` comparator has been implemented:

```
public class CompareKeyframes implements Comparator<Object>{

    @Override
    public int compare(Object k1, Object k2) {

        int order;
        if ( ((Keyframe)k1).getScore() < ((Keyframe)k2).getScore() ) {
            order = +1;
        }
        else if ( ((Keyframe)k1).getScore() > ((Keyframe)k2).getScore() ) {
            order = -1;
        }
        else {
            order = 0;
        }
        return order;
    }
}
```

10.4.8. Validation

Validation is made within the visible page, so the user can control all the keyframes that will be validated. This option avoids forcing the user to validate all keyframes of an asset, he or she can validate as many pages as desired. The user can validate all visible keyframes within the page, or only validate the visible keyframes belonging to a specific semantic shot.



Figure 10.13.: Page validation button

10. Development



Figure 10.14.: Validated page

After validation, every keyframe score is changed to 100% as it is considered a hand labeled instance to be used by the trainer, so it is added to the correspondent semantic shot training vector to update the semantic shot tabs. A green border is also applied to the image by setting a CSS style.

Semantic shot validation within the same page is provided by pressing the validation button inside each of the semantic shot panels. The following figure 10.16 shows an example of shot validation used when other semantic shots need too much errors to be fixed.



Figure 10.15.: Validated semantic shot

10.4.9. Tag as best semantic shot keyframes

The tab panels corresponding to the already validated keyframes show all the keyframes contained in each semantic shot of the TrainingVector. A Scroller object is contained inside the tab so all keyframes can be shown independently of their number.

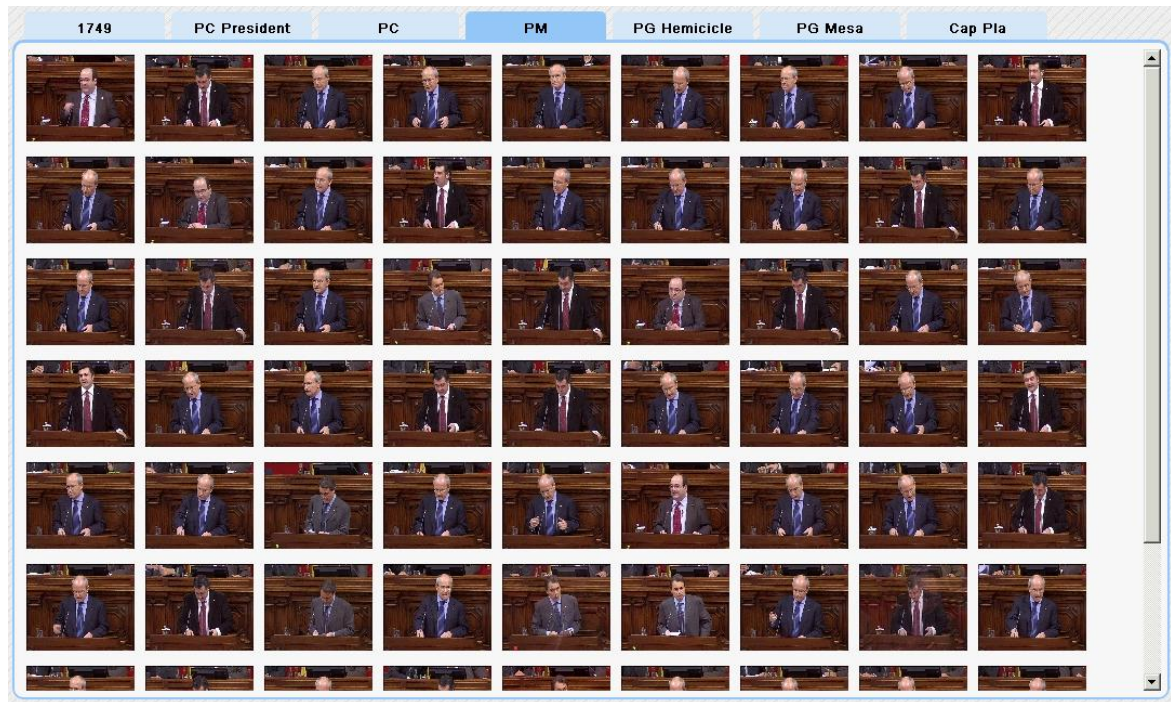


Figure 10.16.: Semantic shot keyframes panel

A handler event on each keyframe allows keyframe selection, changing the keyframe CSS style by adding a red border in order to allow the user to see which keyframes have been selected. The keyframe annotation attribute is also changed from false to true.

To deselect a keyframe the image has to be clicked again, then the red border style is removed and keyframe annotation attribute is set back to false.



Figure 10.17.: Quality keyframes annotation

10.4.10. Dynamic pagination

One of the features the user required was a pagination menu. The selected page and the total number of pages have to be stored to make it possible. The main problem about

pagination programming on this interface is the constant fluctuation of the total number of pages.

A method that computes the total number of pages has been developed and called when an event susceptible to change the number of pages occurs. The following list explains which events can change the total number of pages and why.

- **On detection response:** When a detection is received, the keyframe is added to the vector which semantic shot is the same as the detected one. As it can be seen, there is no way to estimate how many keyframes will belong to each semantic shot, so pages have to be added in a dynamic way.
- **On drag end event:** When there is a drag and drop operation one or some of the keyframes are removed from a semantic shot and added to another. This event can also make increase or decrease the total number of pages.
- **On setting a new minimum score:** When selecting a new minimum score some of the clutter keyframes will be added to the other semantic shots and vice versa. After rearranging all semantic shots the total number of pages has to be computed again.
- **On hide validation event:** When the “hide validated keyframes” option is selected only the non-validated keyframes are shown, so pagination has to be modified again as it would decrease when having enough validations. After hiding validation the current page is set to the first page so the user can continue validating all non-validated keyframes.
- **On validation event:** When validating keyframes, as the first position of each semantic shot is set to be a validated keyframe on each page, semantic shots with at least one validation (drag and drop included) will only show 10 keyframes whereas semantic shots with no validations will show 11 keyframes per page.



Figure 10.18.: Pagination menu

10.4.11. Progress bar

The progress bar layout has been done with two `HorizontalPanel` objects, one inside the other, dynamically changing the width of the inner panel and applying a CSS style in order to show the progress.

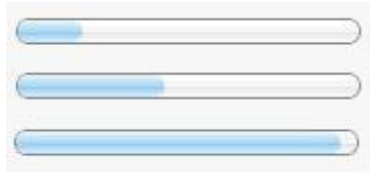


Figure 10.19.: Progress bar showing different percentages

To compute the progress percentage it has been set a counter on each detection response and used the total number of keyframes within that asset.

10.4.12. Tooltips

The tooltips have been implemented using the default tooltip system of each navigator. This can be done by adding a title to the thumbnail. The title for each keyframe includes:

- Keyframe timecode
- Keyframe detection score

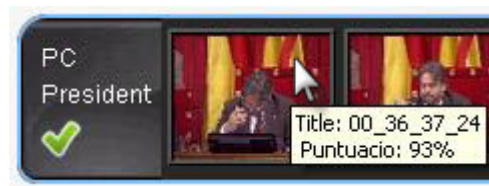


Figure 10.20.: Tooltip showing the keyframe's title and score

This feature keeps the interface simple by not adding too much visual information on the panels and letting the user to consult it in an easy way whenever they want to with just moving the mouse over a specific keyframe.

11. Evaluation

This chapter covers the first GUI impressions and a screen shot showing the final result of all developed features explained in the previous chapter.

11.1. GUI results

This Graphical User Interface has been designed and developed to provide a good user experience based on:

- Simplicity
- Flexibility
- Efficiency
- Completeness

The interface works properly in all tested browsers, which include: Google Chrome, Mozilla, Internet Explorer 8 and Safari.

Figure 11.2 shows the GUI appearance after choosing the parliament domain the asset number 1749.

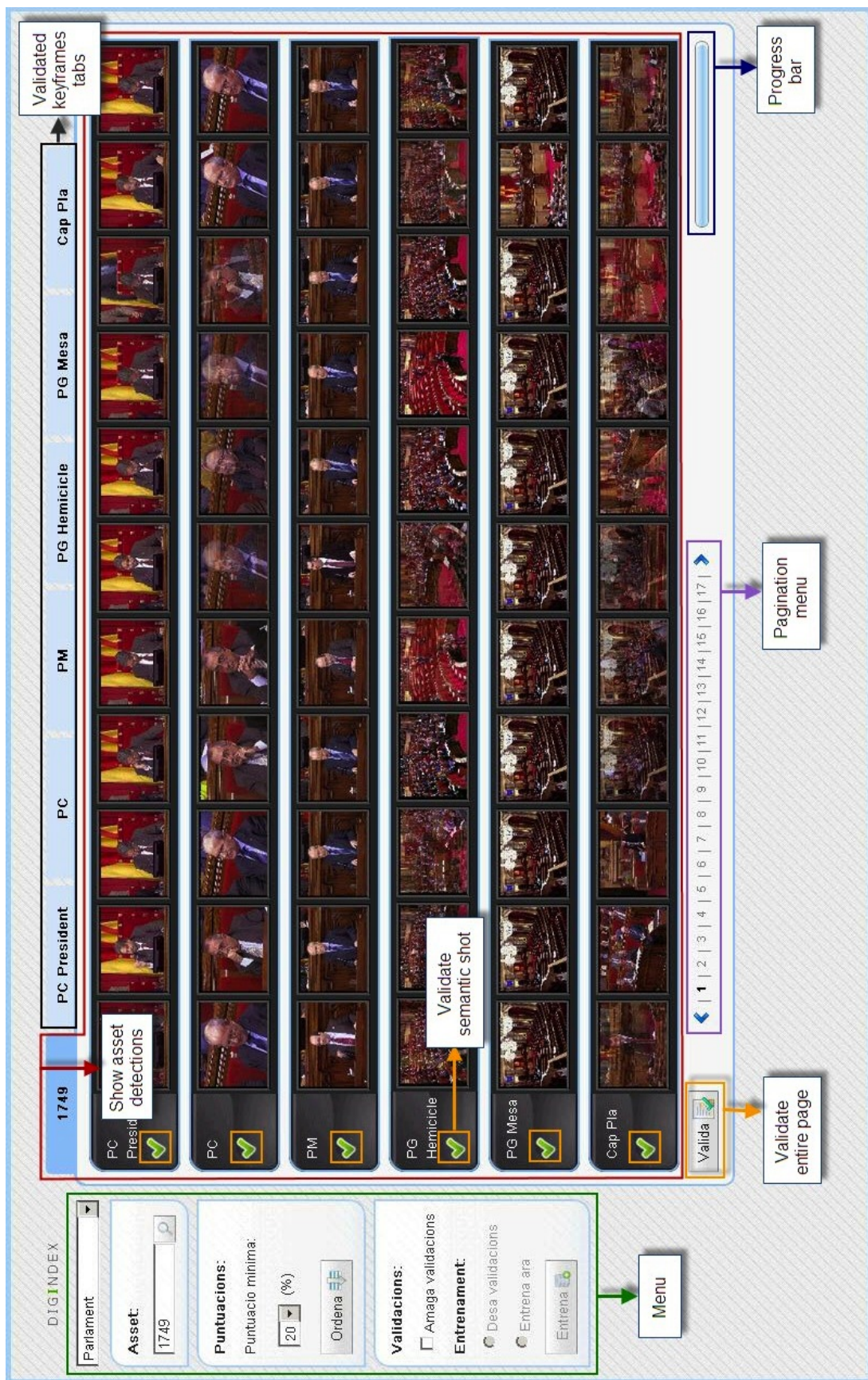


Figure 11.1.: GUI layout

Part III.

Conclusions

12. Compliance of the requirements

This project has been developed in full compliance with the requirements stated by both the university and the company. The next two sections explain the achievements and contributions made in the classifier and the interface.

12.1. Classifier

At the UPC the development of a Soccer match and a Catalan Parliament classifier was required. All listed semantic shots for each domain works fine and the performance of the classifiers has exceeded expectations: 0.97 and 0.86 of FB measure in the Parliament and soccer matches respectively.

The most remarkable contribution of this project has been providing a structure to work with multiclass classifiers, specially ones with a clutter class:

- Since now all annotations had to be done independently for each class and there was no way of labeling all positive instances of a class as negative instances of the other classes.
- There was no way the multiclass classifier could be evaluated in terms of precision and record, only the confusion matrix was provided. I have implemented new metrics to compute precision, recall, F1 and FB measure out of a confusion matrix for each class and the whole classifier.
- Beyond evaluating the classifier performance for both domains Soccer and the Parliament as required, I have also provided the UPSeek department with a generic tool for automatically find the optimum parameters for a given classifier. This is very handfull as optimum parameters can only be known through experiments.
- I have also developed a new dataset partition method, repeated random sub-sampling, which has been used for the experiments.

12.2. Graphical User Interface

The first requirement the CCMA demanded was developing an interface that could be integrated into the Digion. The interface has been developed using the same version of the Google Web Toolkit as the Digion: GWT 2.2.0. The whole interface has been designed using strictly GWT panels and widgets, avoiding other GWT-based frameworks, so it is 100% compatible with the Digion.

The GUI is easy to use but complete at the same time, providing all necessary information the user may need as image tooltips a progress bar and a complete pagination menu. It also has enough flexibility to allow working in different ways so it easily adapts to each user and situation. The efficiency is also a strength of this GUI, as it provides real time experience with drag and drop and provide other tools to help fastening work while giving more accuracy.

The most innovative and difficult part of the interface has been the implementation of draggable and droppable keyframes in an efficient way so the user has a real time drag and drop experience. This task had never been done before in the company and the project team has shown interest on implementing it in a near future. The handlers related to a drag and drop events have also been essential to keep the designed interface, as all the panels had to keep the same columns of keyframes after dragging and dropping.

The tools provided to work with the detections as sorting by detection score, changing the minimum score or validating either the entire page or just a specific semantic shot offers different ways of working, so the user can choose the tools that better suits his or her needs at each moment. The required level of flexibility has been provided.

The dynamism of the interface, allowing different panels and tabs depending on the domain and the dynamic pagination depending on many features such as number of detections on each panel, drag and drop events or other menu options as hiding validations has been a must to suit the interface needs. This makes it suitable for any domain or multiclass classification application, so the requirement of developing a generic interface has also been accomplished.

The only problem is that the interface still cannot use the training web service as the UPSeek team still doesn't know how to parse the JSON message with their server API in a proper way.

13. Further work

At this point, the first priority for the company is having the training system working by implementing the JSON parser at the UPSeek. Another priority is efficiency as the CCMA wants to speed the overall process. The tool has to be faster to suit their needs and the UPC will have to work on that. Another problem to solved are the errors that generate the internal queries to the Fedora Commons database and that make some detections fail.

The system as it works now, only computes the optimum parameters when bulding the first model. The automatic classifier's parameters selection developed for evaluating the classifier is interesting for further application as the optimum parameters depends on the labeled instances used for training and, as they increase when retraining the system, optimum parameters might change too. So when retraining the classifier, the new optimum parameters would have to be found before generating the new model.

The automatic evaluation of the classifier could also be used to determine wether the classifier improves by adding new labeled instances (when comparing results with the previous model performance) in order to avoid the unnecessary growth of the training dataset, which increases the time spent on generating the new model and decreases storage capacity.

Some further work has to be done in the Digation. As the responsible for the Documentation Department expressed, when showing the keyframes within an asset, it would be helpful for them to include the possibility to visualize them rearranged in terms of semantic shots if that asset has been validated. A new drop box would also need to be added on the main searching page to let the user search by a specific semantic shot within a textual search.

Finally, to match the way metadata is created by the Documentation Department at TV3, it would be required a temporal analysis of the asset as for the story board each semantic shot needs a start and an end time code. Independent keyframe based systems just give excessive and duplicated information. Having temporal analysis would also provide the possibility of detecting other semantic shots such as replay shots or others involving movement or still objects.

13.1. Evolving to an intelligent indexing tool

I believe this system can have many applications to provide a complete intelligent indexing tool. It would be interesting to use the semantic shot information to make decisions about what to annotate. Here there are some examples of decisions that could be done depending on the semantic shots defined for this project:

- When a banner is detected in the soccer domain, the Digiton's text detector could extract the text on the banner and add it as metadata.
- When detecting a close up or a medium shot (independently of the domain) it could be used a face detector, a tool the CCMA still does not use but is interested in, to identify the person in that shot and to avoid using the detector with images that are not of interest or where no faces or too much faces are shown.
- When detecting a medium shot at the parliament (which means a parliamentarian is standing up giving a speech), beyond using the face detector, another interesting tool to develop would be one that could transcribe voice to text.

Possibilities are endless. I hope they find the way to exploit all the possibilities the developed tool can offer when using it along with other technologies.

Bibliography

- [1] Ramón Salla; “Aplicació rica d’internet per a la consulta amb text i imatge al repositori de vídeos de la Corporació Catalana de Mitjans Audiovisuals”; ETSTB 2009; <http://hdl.handle.net/2099.1/8766>
- [2] Pia Muñoz; “Extensió d’una interfície de cerca d’imatges a les consultes amb regions”; Escola d’Enginyeria de Terrassa (EET) 2010; http://gps-tsc.upc.es/imatge/_Xgiro/teaching/thesis/2009-2010/PiaMunoz/memoria.pdf
- [3] Mónica Alfaro; “Reordenació i agrupament d’imatges resultants d’una cerca de vídeo”; Escola d’Enginyeria de Terrassa (EET) 2011; <http://upcommons.upc.edu/pfc/bitstream/2099.1/11106/1/memoria.pdf>
- [4] Activa Multimèdia; “DigitonSuite”; <http://www.activamultimedia.com>
- [5] Wang et al (2006); “An Efficient Automatic Video Shot Size Annotation Scheme”; <http://www.springerlink.com/content/e25146466j4166rl/>
- [6] Duan Xu et al (2003); “A Unified Framework for Semantic Shot Representation of Sports Video”; <http://www1.i2r.a-star.edu.sg/~tian/ConferencePapers/SPIE/SPIE03-SportsVideo-DuanXuTian.pdf>
- [7] a b Olson, David L.; Delen, Dursun “Advanced Data Mining Techniques” Springer; 1 edition (February 1, 2008), page 138, ISBN 3540769161.
- [8] van Rijsbergen, C. J. (1979). Information Retrieval (2nd ed.). Butterworth.
- [9] J. Shao, J. Am. Stat. Assoc., 88 (1993) 486-494.
- [10] Q.-S. Xu and Y.-Z. Liang, Chemom. Intell. Lab. Syst., 56 (2001) 1-11.
- [11] M. Clementi, S. Clementi, M. Fornaciari, F. Orlandi and B. Romano, J. Chemometrics, 15 (2001) 397-404.

Bibliography

- [12] Carles Ventura; “Image-Based Query by Example Using MPEG-7 Visual Descriptors”; Escola Tècnica Superior de Telecomunicació de Barcelona 2010; <http://upcommons.upc.edu/pfc/handle/2099.1/9453>
- [13] Mireia Gimeno; “Interfície gràfica d’usuari per a l’avaluació de classificadors d’imatges”; Escola d’Enginyeria de Terrassa 2010; http://gps-tsc.upc.es/imatge/_Xgiro/teaching/thesis/2009-2010/MireiaGimeno/memoria.pdf
- [14] Xavier Giro-i-Nieto, Neus Camps, Ferran Marques,; “GAT, a Graphical Annotation Tool for semantic regions” ; Multimedia Tools and Applications 2009; <http://dx.doi.org/10.1007/s11042-009-0389-2>
- [15] Delco Perez; 2006; <http://hdl.handle.net/2099.1/3855>
- [16] Heyer Lurie J., Kruglyak Semyon, Yoosep Shibu; “Exploring Expression Data: Identification and Analysis of Coexpressed Genes”; Heyer-QT 1999; <http://genome.cshlp.org/content/9/11/1106.abstract>
- [17] Khristina Bakerkina; 2009; <http://hdl.handle.net/2099.1/7925>
- [18] MacMillan, K, Droettboom, M, and Fujinaga; ”Gamera: A Python-based Toolkit for Structured Document Recognition”; International Python Conference 2001; http://dkc.jhu.edu/gamera/papers/gamera_python_2002/gamera.html
- [19] Damian Borth, Adrian Ulges, Thomas M. Breuel; “Lookapp- Intreactive Construction of Web-based Concept Detectors” - ICMR 2011; ISBN: 978-1-4503-0336-1 doi:10.1145/1991996.1992062.
- [20] M. Bertini, A. Del Bimbo, G. Ioannidis, A. Stan, E. Bijk; “A flexible environment for multimedia management and publishing”; ICMR 2011; ISBN: 978-1-4503-0336-1 doi:10.1145/1991996.1992072.

A. Classifier parameters analysis

This appendix makes an overall analysis of the effects that the minimum score, the maximum radius and the minimum number of elements has on the classifier performance, so the optimum values for each parameter can be understood.

This analysis also highlights the repercussions that inappropriate parameter values have on the F1 measure and thus the importance of using a methodology to determine the optimum values.

A.1. Minimum score

When observing the effects on increasing the minimum score there are two positive effects: the recall of the clutter class (class 7) and the precision of all classes increase too.

On the other hand, recall of classes which detections are scored under the minimum score will decrease as those detections would be considered as class7. This can be seen on classes 3, 5 and 6 on figure A.1.

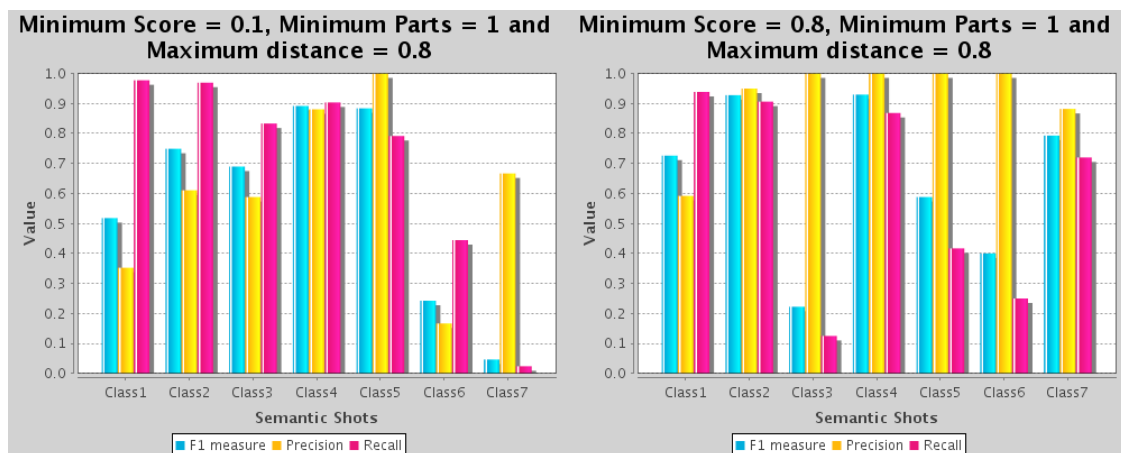


Figure A.1.: 0.1 vs 0.8 minimum score on Soccer matches classifier

A. Classifier parameters analysis

Also notice that the precision of a class only increases if there are detections with a higher score than the minimum score, otherwise no detections would be assigned to that class and the performance of the classifier will decrease. This can be seen on figure A.2 when setting a 0.9 minimum score.

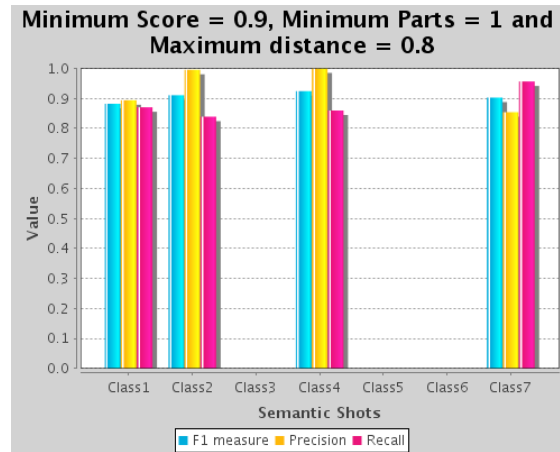


Figure A.2.: 0.9 minimum score on Soccer matches classifier

A.2. Minimum number of elements and maximum radius

Figure A.3 shows the maximum radius versus the minimum number of elements in Soccer Matches with 0.7 minimum score. It has one ribbon for each maximum radius computed and shows how F1 measure changes depending on the minimum number of elements.

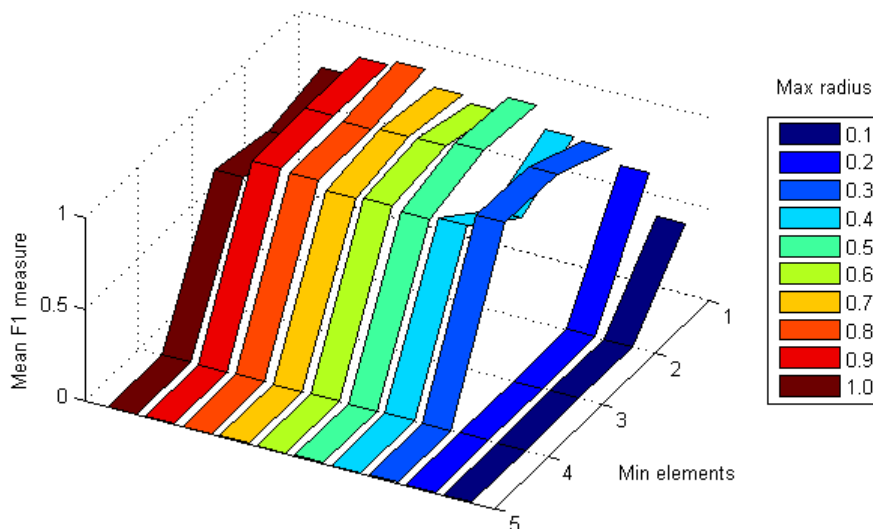


Figure A.3.: Maximum radius ribbons at 0.7 minimum score

A.2. Minimum number of elements and maximum radius

It can be seen that the F1 measure tends to decrease when the minimum number of elements increases. It can also be observed that ribbons corresponding to 0.1 and 0.2 maximum radius have a lower F1 measure than the other ones, as the shorter the distance, less elements would be contained in it.

The optimum value depends on the images that have been labeled, their number and their similarity. The more instances labeled and the more similar, the higher minimum number of elements can be possible.

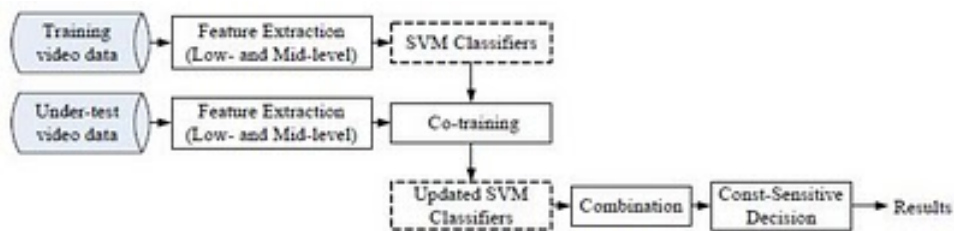
B. Contributions to Bitsearch blog

This appendix collects all posts that I have been made in the Bitsearch blog about the project. Bitsearch can be visited at <http://bitsearch.blogspot.com/>.

B.1. Automatic video shot type identification by Wang et al

My name is Eli Carcel and this will be my first contribution to this blog. As I am now starting my bachelor thesis on keyframe-based shot identification I have been studying the state of the art. I found a related paper called "Automatic Video Shot Size Annotation Scheme by Wang et al (2006)" which I will briefly describe.

The aim of the project is recognizing an image shot type pattern, assuming there are three possible categories: Close-up, Medium shot and Long shot. The project steps include: feature extraction, training, classification and decision.



Process

The first step is feature extraction where two feature vectors are created, one corresponding to low level features such as color, edge and texture and the other one to mid level features involving regions and its features.

Semi-supervised learning is needed in order to obtain valid data to train the system. Co-training can help boosting the training process by offering semi-supervised annotation.

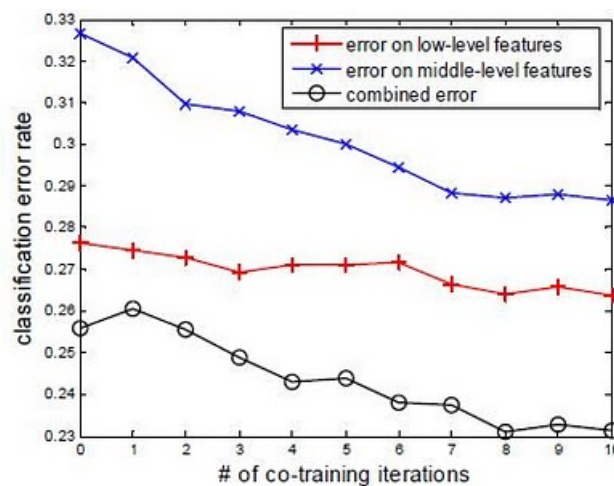
B.2. Supervised, unsupervised and semi-supervised learning

Each vector of features is sent to a different classifier (low level and mid level) to decide which shot size suits best the image among the three existent classes. A combination of both classifiers' results is used in order to improve its efficiency. Cost-sensitive decisions for misclassifications are calculated to provide a better cost minimization based classifier.

Experimental results

Tests are done using 20 hours of video extracted from home video camcorders and then divided into 1000 shots according to timestamps. As in this stage some shots may include images with several shot sizes, each shot is divided into 4000 sub-shots, assuming now identical shot size within a sub-shot.

All results are the average of 10 runs, where a run is an iteration done by using 20% of the samples randomly selected as training data and the other 80% to test the system.



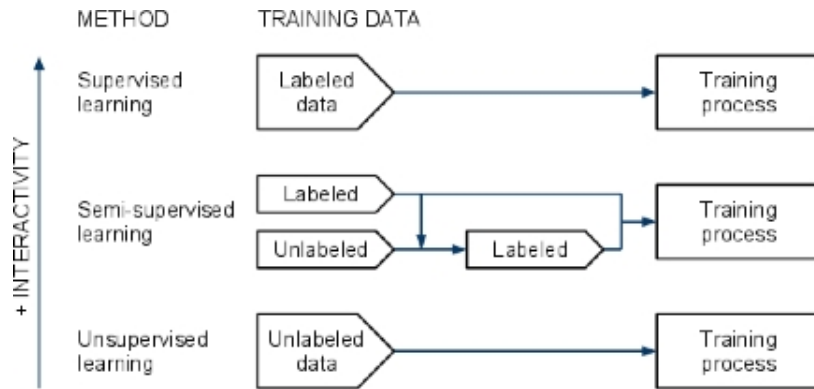
The paper shows how low level descriptors work better in shot type identification, though it can be pursued a better performance using a combined low level and mid level decision.

B.2. Supervised, unsupervised and semi-supervised learning

In this post I will discuss the pros and cons of some machine learning procedures and how would they suit shot type detection at the CCMA (Catalan Broadcasting Corporation). I will focus on three main categories: supervised, unsupervised and semi-supervised learning.

B. Contributions to Bitsearch blog

The difference between these types of learning is whether the training data has been hand-labeled or not to generate the classifier's output. Supervised learning assumes that a set of training data has been provided. On the other hand, unsupervised learning has no labeled data. A combination of both labeled and unlabeled data results in semi-supervised learning.



What does that imply?

In supervised learning, a qualified person would properly label by hand the set of instances to be used for training. This means professionals at the documentation department would be required for this task. The more labeled instances the classifier gets the more precise will be the output, so the amount of hand labeling needed to reach a desired quality is certainly time-consuming. The advantage is that the system can be controlled better: shot types can be defined and it can be selected which images are of interest. For example, pictures with no people can be avoided by labeling them as “not relevant shot type” so not relevant shot type pictures will not be shown on a search.

Unsupervised learning, having no labeled data, attempts to find out similar patterns in the data to determine the output. This type of learning needs nobody for the training process but this also means no interactivity: the system has no clue on which shot types are of CCMA's interest and will define different classes clustering similar data depending on the visual content. This kind of learning wouldn't suit the CCMA needs as different shot types have to be defined depending on the context: a soccer close-up shot would have a different name than a parliament close-up shot.

Finally, semi-supervised learning is actually a supervised method that avoids labeling a large number of instances. This is done by using some of the labeled data to help the classifier labeling the unlabeled data. Then, this automatic labeled data is also used by the training process. Another supervised method that helps mining labeled data is called

active learning. Basically, it decides which data should be labeled to improve the classifier performance with less data. These two options are really interesting, as they have the benefits of both supervised and unsupervised learning: interactivity and taking advantage of unlabeled data. With a few labeled instances and the great amount of unlabeled images at our disposal this system could perform well for shot type detection.

B.3. Measures out of a confusion matrix

A confusion matrix is used in supervised learning for comparing the outcome classification of an item with the desired classification. Each row of the matrix represents the instances that have been automatically predicted in a class, while each column represents the hand-labeled instances in a class. This matrix is useful for observing which classes have mis-identified items as other classes.

		Automatic		
		class1	class2	class3
Manual	class1	12	1	0
	class2	3	13	0
	class3	0	0	20

Precision and recall:

Precision or specificity is a mesasure of the ability of a system to present only relevant instances. It measures the exactness or fidelity of the system.

$$Precision = \frac{\text{number of relevant instances retrieved}}{\text{total number of instances retrieved}}$$

Recall or sensitivity is a mesasure of the ability of a system to present all relevant instances, so it is used for evaluating the completeness of results.

$$Recall = \frac{\text{number of relevant instances retrieved}}{\text{number of relevant instances in the collection}}$$

B. Contributions to Bitsearch blog

It is difficult to compare different systems in terms of precision and recall, as both measures are independent. Moreover, when recall increases precision tends to decrease: as more relevant instances are retrieved, the more nonrelevant instances are retrieved.

F and F β measure:

F-measure considers both precision and recall providing a single measurement for a system avoiding having two independent measures.

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

In order to give different weights to precision and recall, the F-measure was derived so that F β measures the effectiveness of retrieval with respect to a user who attaches β times as much importance to recall as precision.

$$F_{\beta} = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

F β measure of a confusion matrix:

For my bachelor thesis I have had to write a function capable to compute the F β measure for a given confusion matrix. This is the equation that has been derived from the explained measures:

$$MF_{\beta}(i) = \frac{(1 + \beta^2)}{N} \cdot \sum_{n=1}^N \frac{\frac{x(i,i)}{x(i,n)} \cdot \frac{x(i,i)}{x(n,i)}}{\beta^2 \cdot \frac{x(i,i)}{x(i,n)} + \frac{x(i,i)}{x(n,i)}}$$

Where "i" represents the class and "N" is the total number of classes.

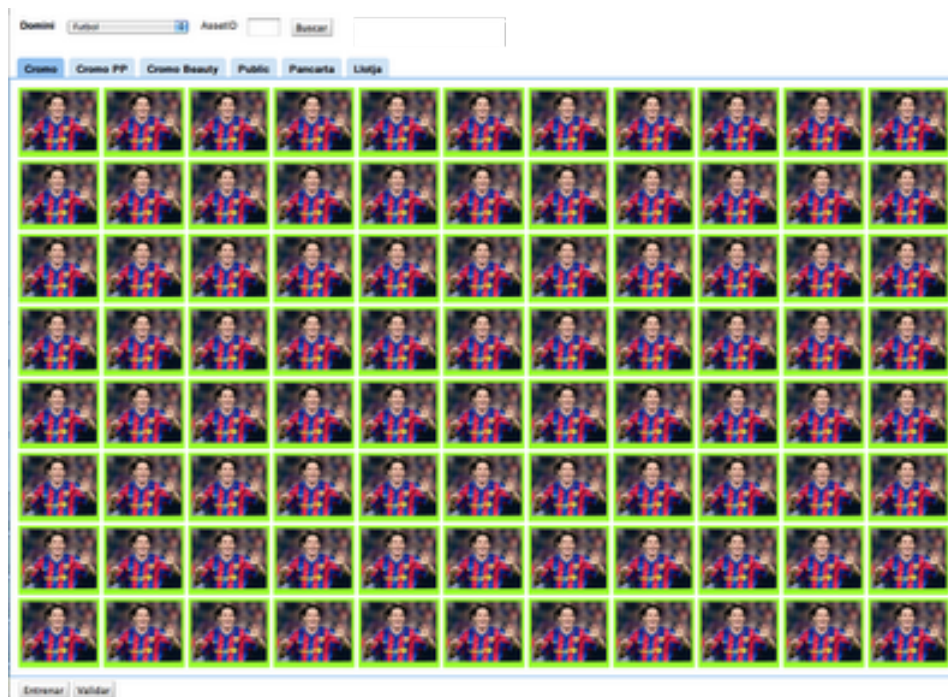
B.4. Web interface for shot type detection

In this post I will explain the evolution of the web interface I have to develop for my bachelor thesis, which pursues semi-supervised key-frame shot type identification. This interface will be used to re-train a shot type classifier (already trained with manual labeled examples) for a given domain by validating or correcting its automatic detections and then adding them to the trainer to generate a new model.

So for a given domain and an asset (group of keyframes of that domain) the interface will have to show the automatic shot type identification and provide the necessary tools for validating the automatic detections or correcting them if necessary.

After one month of work, two different layout proposals were presented to the Documentation Department at the CCMA in order to keep developing the interface with the structure that better suit their needs, these are the following ones:

1st Proposal:

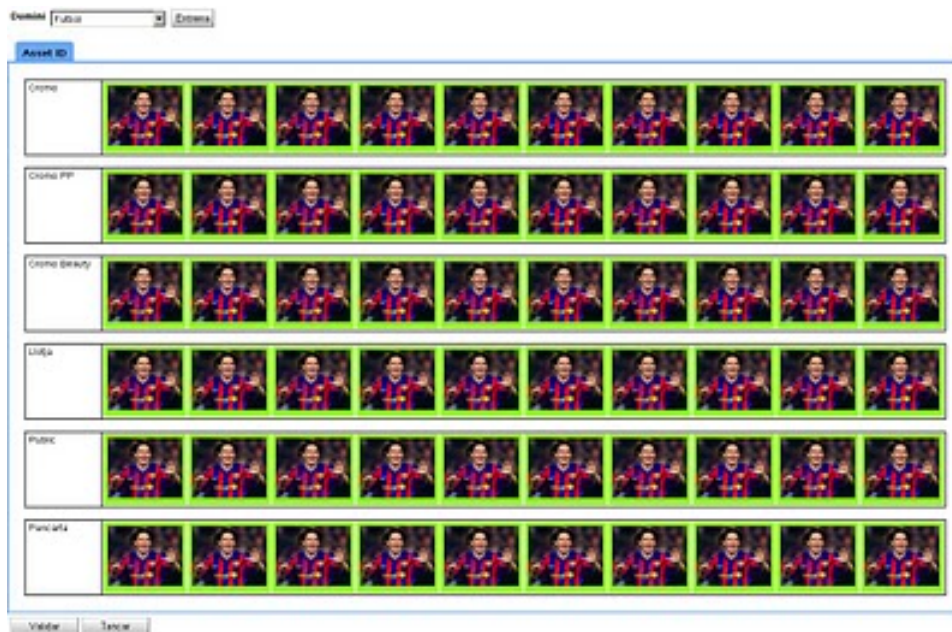


- For each domain we have different tabs for each shot type.
- Each tab contains all the detected keyframes for that shot type.

B. Contributions to Bitsearch blog

- The number of keyframes on each tab is fixed, the variable thing is the number of pages for each tab, which depend on the quantity of keyframes detected for that shot-type.
- Keyframes have to be labeled as positive (detection is ok) or negative (detection is wrong)
- Besides labeling as negative a keyframe the information of the right shot-type has also to be provided.
- Only one shot-type can be seen at once.
- To open another asset the already opened one has to be closed, cannot manage working with different assets at the same time.

2nd Proposal:

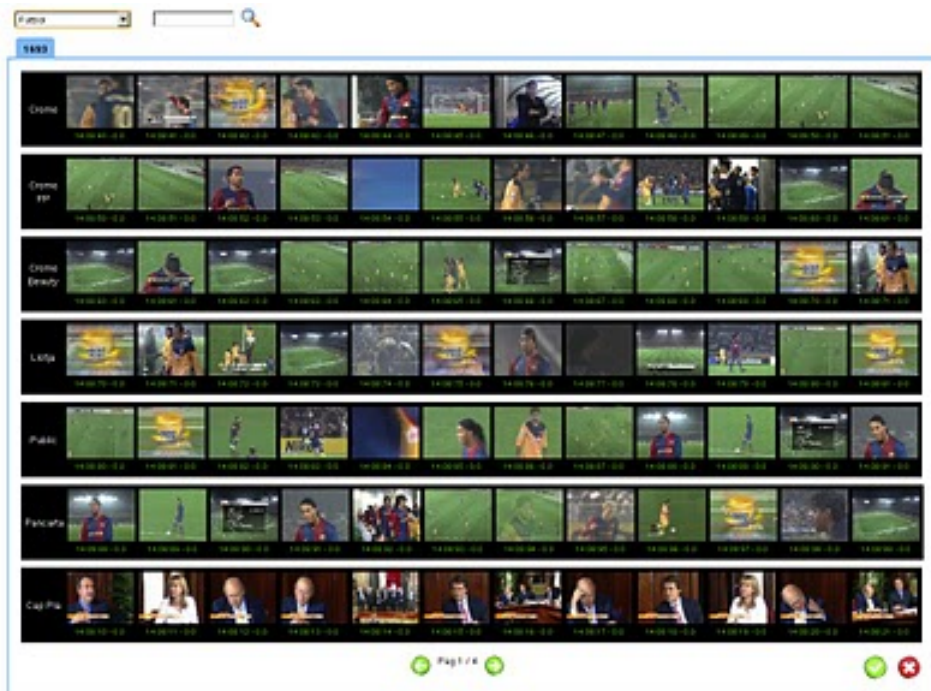


- For each domain we have different tabs for each asset, so simultaneous working with several assets is supported.
- For each asset tab there are as many keyframe rows as shot types in that domain. So all shot types can be seen at once.

- The columns of each row are fixed for all shot types. As not all shot types have the same keyframes detected and pagination is shared by all shot-types, some of the rows can run out of keyframes.
- This layout makes it suitable for drag and drop functionalities to easily correct a keyframe shot-type detection, so there is no need of labeling any keyframe as negative, just drop it to the right place.
- Once all keyframes are in the right place the page can be entirely validated.
- As when validating there are examples of all shot types within a domain, at training the classifier would have similar number of examples for each one so it will improve in a more homogenic manner (a class will not be undertrained as it can happen with the first proposal).

Current interface:

Because of its functionality and its visual component, the 2nd proposal was chosen by the CCMA with additional requirements as showing the title and the detection score for each keyframe. This is its actual appearance



B.5. Dataset generation

In this post I will discuss two different techniques for dataset generation that have been studied and implemented for my project on semantic shot detection: modified K-fold cross-validation and repeated random sub-sampling

For testing a multiclass classifier, the annotated positive and negative instances of all classes have to be split into a trainset and a testset so the trainer and the detector use different sample data for the experiment. Because of the architecture of the classifier used in my project, based on binary classifiers, each class annotations are partitioned in order to train and test each semantic shot classifier.

In this case, as there were few labeled instances, 80% of the each class sample data has been used to train its classifier while the 20% left has been used for detection. As there are classes with a few positive instances and much more negative instances (all the positive instances of the other classes), each percentage has been applied separately to positive and negative instances, ensuring at least positive instances to train the system.

All instances belonging to the trainset clutter class have been deleted so the trainer does not include the class when generating the model, but the detector can measure its performance when detecting.

Repeated random sub-sampling

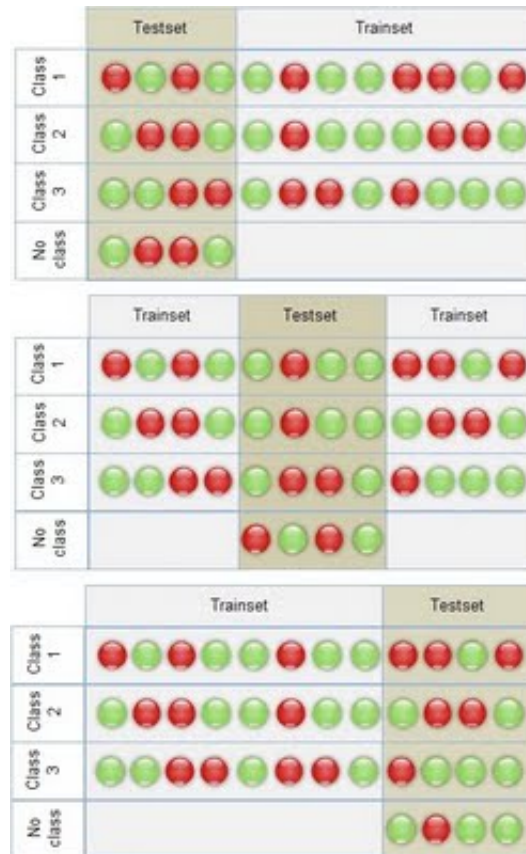
Random sub-sampling consists on sorting the data in an aleatory order and then dividing it into the dataset and trainset for the given percentatges. For each iteration a single trainset and dataset is build. The algorithm has been applied separately to each class to train each classifier.

Modified K-fold cross-validation

The K-fold cross-validation algorithm generates K different data subsets, making sure all data is used for both training and detection for each iteration.

As a modification of the K-fold cross-validation, where data ara partitioned into K equal parts and the ratio between the sizes of the different subsets is not defined by the user, the parameter K has been calculated automaticaly to build the testset and trainset for the given trainset percentage, allowing more flexibility.

This procedure is repeated K times and the predictions of the K dataset are averaged. The whole procedure can be done several times with aleatory data order for each iteration. The following picture illustrates one iteration of the process:



Conclusions

Random-subsampling does not ensure all instances are both used for training and detection, but more iterations can be done with less time, as random sub-sampling represents 1-fold iteration of our modified cross-validation. Predictions using this method give a realistic estimation of the classifier performance with external validation data, while cross-validation usually gives overly optimistic estimations.

Because of the time cost of the K -fold cross-validation, the diversity of the datasets that already offers random sub-sampling and its realistic estimation, the repeated random sub-sampling algorithm has been chosen for the experiment dataset builder.