SEVENTH FRAMEWORK PROGRAMME

SignSpeak

Scientific understanding and vision-based technological development for continuous sign language recognition and translation

Report on the Preliminary Collection of Videos Segmented Mainly by Low-level Cues

Minor Deliverable D.3.1.M12

Release version: V1.0

Grant Agreement Number 231424

Small or medium-scale focused research project (STREP)
FP7-ICT-2007-3. Cognitive Systems, Interaction, Robotics

Project start date: 1 April 2009
Project duration: 36 months

Dissemination Level		
PU	Public (can be made available outside of SignSpeak Consortium without restrictions)	
RE	Restricted to SignSpeak Programme participants and a specified group outside of SignSpeak consortium	
IN	SignSpeak Internal (only available to (all) SignSpeak programme participants)	X
LI	SignSpeak Limited (only available to a specified subset of SignSpeak programme participant)	
Distribution List (only for RE or LI documents)		

0 General Information

0.1 Document Information

Title	Preliminary Collection of Videos Segmented by Low-level Cues - Report
Type	Minor Deliverable
Ref	D.3.1.M12
Target Version	V.1.0
Current Version	V.1.0
File	SignSpeak.D.3.1.M12.doc
Author(s)	Jaume Vergés-Llahí
Reviewer(s)	Gregorio Martínez Ruíz (CRIC)
Approver(s)	Gregorio Martínez Ruíz (CRIC)
Approval date	
Release date	

0.2 Document Scope

This document accompanies Deliverable D.3.1, which gathers a preliminary collection of video segmented by low-level cues. It describes the algorithms developed and implemented as part of Task 3.1, concerned with the spatio-temporal feature extraction as part of the multi-modal visual analysis in WP3.

0.3 Document Content

1 Overview

The primary objective of WP3 is the development of methods for video analysis to produce a stream of features which can be used by the sign language recognition and translation methods developed under WP4 and WP5. With respect to the present Deliverable D.3.1, it is concerned with managing video sequences, performing low-level operations on the images, detecting and tracking candidate regions where to locate hands within the images, and extracting distinctive spatio-temporal features to describe such regions in order to locate the hands in a more discriminative way at a posterior stage of the development of the project.

The logical order of the developments which has been followed so far to perform the work in Taks 3.1. consists in capturing the video data, preprocessing the images, detecting candidate regions which could correspond to hands based on motion and skin color, extracting a distinctive description and identifying the regions with hands in order to locate them and constructing a description of their motion and configuration, which will be used afterwards in the recognition and translation of the sign language.

In the following paragraphs each of the techniques employed to accomplish the previously mentioned subtasks will be succinctly described in relation to the objectives of Task 3.1. The description will account for the interest in its use, the basics of its functioning, pointing out the work that has already been accomplished and those which will be endeavoured in the short future. In the Annex section, some algorithms have been described with further detail.

2 Objectives of the Task 3.1

This Deliverable is concerned with the Task 3.1 and corresponds to the description of the preliminary collection of videos segmented by low-level cues. Specifically, work in Task3.1 is devoted to the extraction of spatio-temporal features for the description of image content, for their integration into D.3.2.

The objectives are that the spatio-temporal feature extraction from raw video data will be robust and self-adapting to the changing ambient conditions, taking into account the independence with respect to the signer, background clutter and illumination. Besides, it is a design requirement that the signers will not wear any kind of artificial marker, sensors, or gloves, which therefore facilitates a more natural signing scenario.

In order to carry out the objectives of the present Task, the work that must be fulfilled has been divided into two main tasks described as follows:

- ***Low-level image and video processing*** includes operations such as filtering, keypoint extraction, and tracking, etc. This will mostly rely on established techniques, but the work carried out in this Task has been necessary to identify, adapt, and integrate those that are best suited to our objectives. This is illustrated in more details in Section 3 later in this document.

- ***Selection and construction of distinctive features*** for *recognition* (WP4) and *translation* (WP5). Human sign language interpreters rely on parameters that can be explicitly described in everyday terms, such as hand trajectories, finger configurations, and facial expressions. However, for machine recognition and translation, it is likely that there are other powerful cues that are more easily or robustly extracted than these explicit parameters, such as spatio-temporal image statistics or combinations of other cues. This task is concerned with their identification and construction, and making them available to tasks within WP4 and WP5. More details on feature description is Section 4 of this document.

No prior software existed in CRIC concerned with such kind of work and all implementations and developments have been endeavoured from the scratch. This means that a great part of the work carried out so far has consisted in finding, evaluating, and selecting the libraries necessary to obtain the kind of visual information required for our purpose, implementing our own codes, and the initial versions and trials of the operations, which have increased in complexity in the direction of attaining the final goals of the present Task.

At this stage of the work, this deliverable encompasses a series of the modules in a first stage of development encapsulated in C++ classes, described later in this document. During the second year of the project, the deliverables D3.1 and D3.2 will be integrated into a larger prototype for multi-modal visual analysis of video sequences.

3 Low-level Image and Video Processing

The set of low-level image and video processing operations are concerned with the preprocessing of frames in order to remove noise and adapting their data so that they can be used by the rest of functions implemented in other modules of Task 3.1. These operations are packed as member functions that are passed to the class **CProcessVideoBase** which applies them to each frame of the sequence.

The nature of the operations performed spans from filtering, color conversion, removal of regions according to different criteria, such as size, position, shape or color, post-processing of segmentation results by removal of holes, morphological operations, and analysis of the blobs obtained by binarization or segmentation (thresholding, histogram back-projection, or background segmentation) of images. These are basic and standard image processing operations which must be carried out in the video data in order to be used it in later stages of the visual analysis of the videos. Most of them correspond to functions in *OpenCV* and *IVT*, which are completed by providing the necessary auxiliary variables in order to easy their usage and integration.

3.1 Contour Detection

This section is about the algorithm employed in the computation of image contours. The information provided by the contours will be employed afterwards in the generation of a description of hand movement, since it carries structural information of the hand configuration that can be helpful in its recognition, though noisy and variable.

The contours of hands are obtained by applying a Canny filtering on an image that has been previously filtered out from noise. Later, the set of contours obtained are also selected according to their length. So far, a set of basic features corresponding to each contour are able to be computed, such as area, length, and bounding box. Additionally, as it will be mentioned later, the algorithms corresponding to local appearance feature computations, namely, *SURF*, *SIFT*, and *HoG*, have been adapted to compute such features on the points corresponding to such contours, in a way to reinforce the structural and appearance description of the hands.

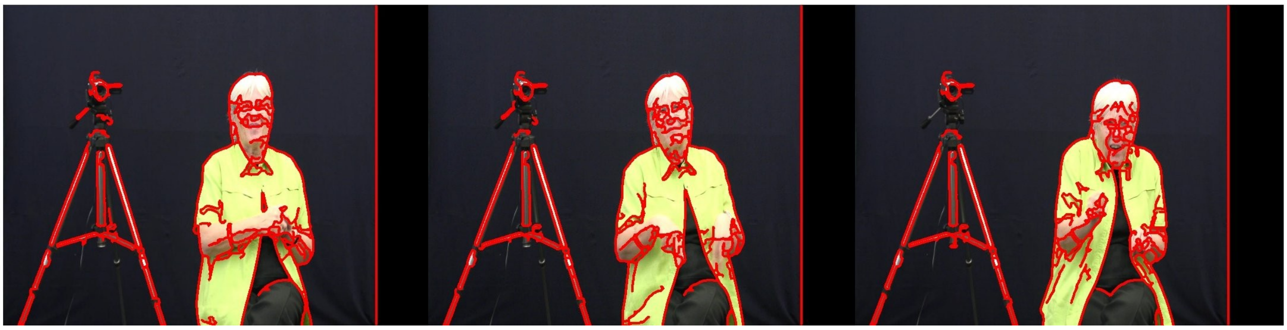

Fig. 1. Some frames showing the contour detection on the video sequence.

3.2 Color and Motion Segmentation

In sign language understanding, the most important part of the meaning is conveyed by hands. Therefore, there must be put a big deal of effort in segmenting not only the position of the hands, but also their configuration in order to obtain higher scores in the recognition and translation of the sign language. Nevertheless, in the first stages of this work, as are they are described hereafter, the main goal is that of isolating the area containing the hands, or at least, a set of likely candidates, which will be tracked and described so as to carry out a more discriminative analysis posteriorly.

The main two cues which are considered in the first stages of this work in order to obtain such candidate regions are motion and skin color. For the first cue, a background segmentation algorithm is taken into account in order to separate pixels corresponding to the moving parts of the body from more static background pixels. For the latter, an algorithm which segments pixels according to their color is employed to detect skin regions. Both are described later in this document.

3.2.1 Background/Foreground Segmentation

The method envisaged in this Task 3.1 for the isolation of objects with a distinctive nature resembling that of hands and face consists in background segmentation algorithm. The reasons for doing this are, first, that the camera mostly looks out on the same static background. The only part which is of interest to us is the moving part of the signers, that is, their faces and hands.

In this Task, the codebook background segmentation algorithm is employed [11] where learning a background model is accomplished by defining boxes by means of an interval over each color axis. These intervals will expand if new background samples fall within the learning limits of the box. If new background samples fall outside of the box and its learning limits, then a new box will be started. In the background difference step, that is, when the image pixels are classified as belonging or not to the background model, by using the threshold values defined by these boxes, a pixel is said to be close enough to a box boundary and then counted as if it were inside the box. The objects left after subtraction are presumably new foreground objects, faces and hands in our case. More details on this algorithm can be found in the Annex 10.1.

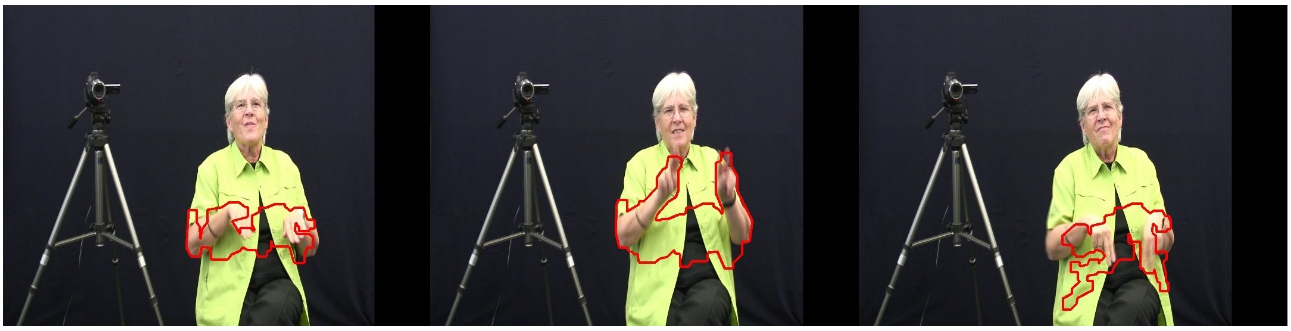

Fig.2. Background/foreground segmentation: Areas in the image with motion appear segmented. These regions may belong both to the signer's limbs and to body parts.

3.2.2 Adaptive Skin Detection

Available skin detection algorithms are either based on static features of the skin colour, or require a significant amount of computation. Such skin detection algorithms are not robust to deal with real-world conditions, like background noise, change of intensity and lighting effects. This situation can be improved by using dynamic features of the skin colour in a sequence of images.

The skin detection algorithm here is based on adaptive hue thresholding and its adaption using a motion detection technique. The skin classifier is based on the hue histogram of skin pixels, and adapts itself to the colour of the skin of the persons in the video sequence. This algorithm has demonstrated improvement in comparison to the static skin detection methods. The first part of the algorithm uses a static skin detector, the global skin detector, that can detect the actual skin pixels with reasonable rate requiring few computational time per pixel. In order to improve the results in a real-world application, there is a second stage with an adaptive threshold that uses motion information to improve the detection of skin colors through the time. The color histograms obtained applying these two thresholds on the images are combined according to a mixing scheme. The resulting color information is used to segment the skin regions out by means of histogram back-projection. More details on this algorithm can be found in the Annex 10.2.

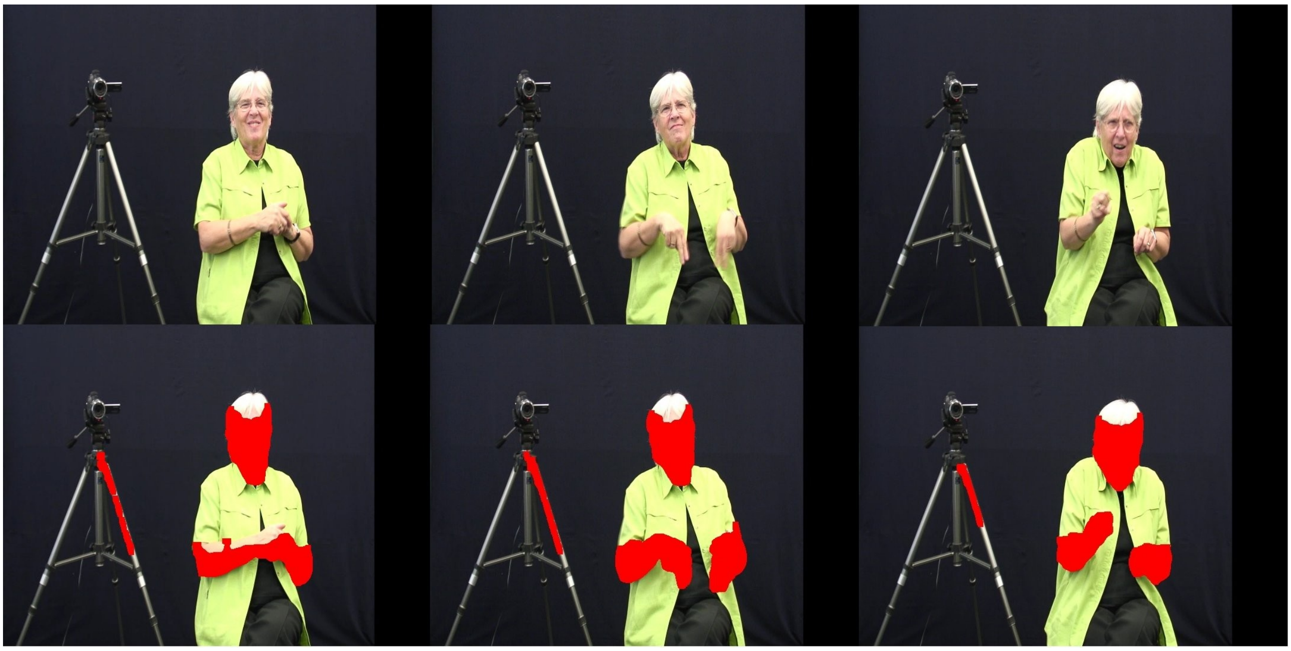

Fig.3. Skin segmentation segmentation: Areas in the image correspond to moving parts of the body which color is similar to that of skin, such face, arms, and handss.

4 Selection and Construction of Features

The second main goal of the Task 3.1 is concerned with the selection and construction of a set of distinctive features for the tasks of *recognition* (WP4) and *translation* (WP5) of the sign language. For automatic recognition and translation, cues other than hand trajectories and finger configuration can be more easily and robustly extracted, such spatio-temporal image statistics or combinations of other cues.

Specifically, the two subtasks which conform the visual analysis of hands are detecting and tracking. Hand detection is difficult due to the great variability in the shape of hands. The fundamental difficulty in extending other approaches standardly used for face detection and tracking do not directly apply due to the highly deformable nature of hands which makes the alignment extremely difficult. Unlike these cases, the positive training samples trimmed to select a hand region, which are usually rectangular boxes, includes an important number of background clutter pixels, qualitatively varying depending on the hand posture, which really influence the training and can make the traditional approaches fail. Discriminative hand detection is usually carried out by learning a set of positive and negative samples of hands in different configuration and later classifying candidate windows from the image by means of a classification algorithm.

In the current approach, prior to the extraction of a set of features for the recognition and translation of sign language, it is necessary to detect and track the group of image regions which likely might contain the signer hands, based on the adaptive skin color detector and a tracking algorithm. Once the hand regions are detected and segmented out, the tracking procedure is carried out on them. Specifically, the *Cam-Shift* and *Mean-Shift* algorithms have been used to perform tracking of the regions obtained from a detection stage, based on skin color and motion cues, as explained above. The reason for using hand detector and tracker are two. First, focusing the process of feature extraction and, as a consequence, speeding up the whole process.

The work so far has been focused on the creation of structures and code which permit the integration of some of the most salient features available in several libraries (OpenCV and IVT). These features consist of **SURF** (*Speeded Up Robust Features*), **SIFT** (*Scale Invariant Feature Transform*), **HoG** (*Gradient of Oriented Histograms*). In addition to them, it has also been incorporated into this code several operations for the detection of interest points, such as Harris detector and **GFT** (*Good Features to Track*). Finally, a representation based on these features named **BoVW** (*Bag of Visual Words*) has specifically been implemented as a mean of describing the visual content of regions enclosing hands.

4.1 Features Descriptors

For any object in an image, interest points on the object can be extracted to provide a feature description of the object. This description, in case of a learning process, can be then used to identify the object when attempting to locate the object in a test image containing many other objects. It is important that the set of features extracted from the training image is robust to changes in image scale, noise, illumination and local geometric distortion to perform reliable recognition.

In this Task, the SIFT[8], SURF[9], and HoG[10] feature descriptors are briefly considered. A further consideration of such features is illustrated in the Annex 10.3. Their computation presents two steps. First, the detection of interest points where to compute feature descriptions of the appearance and, second, the generation of the descriptors themselves. In the available libraries which implement in practice such features, both elements are tightly entangled one to another. Nevertheless, in our implementation these steps have been separated *deliberately* so the set of points where to compute the features can be selected independently using other methods, such as Harris corner detector, GFT, the star point detector, or exhaustively computed on all points in a region.

The aim in Task 3.1. is the extraction of a description of hand by means of spatio-temporal features. However, the features mentioned above are local spatial features. Therefore, it is necessary to relate such features in a certain instant with those computed in the next time step. This task is currently accomplished by matching the sets of features in two consecutive frames. The matching algorithms employed in this work are based on a *nearest neighbor search* in a k-dimensional tree in order to find similar features belonging to two consecutive frames. Some clustering algorithms have also been taken into account. Namely, the *k-Means* and the *Expectation-Minimization* algorithms, which can be used for clustering features obtained directly from video frames, or as part of the computations of the *BoVW*.

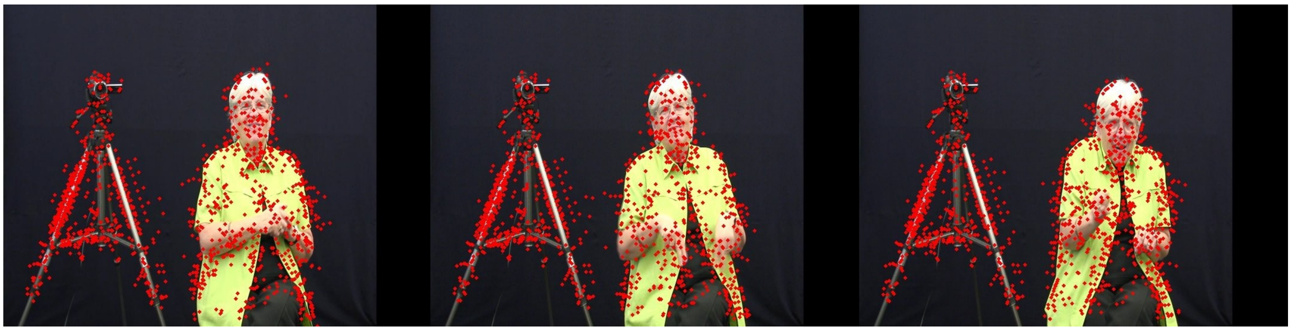

Fig.4. Interest point detection and feature extraction on video sequences.

4.2 Bag of Visual Words

In this section, a brief exposition of the method to describe images based on the structure named *Bag of Visual Words* (BoVW) [4, 7] is shown. This construction is expected to provide a representation of hands discriminative enough to satisfactory perform the tasks of *recognition* and *translation,* based on a set of features as those considered in Section 4.1.

The basic procedure to compute a BoVW consists, first, in obtaining a set of features descriptors from a group of images (corpus), clustering them to obtain a group of representatives (visual words), and then codifying any image (or part of it) as a vector of the presence or frequency of such visual words. This vector forms a BoVW. More details are given in Annex 10.4.

5 Color-based Hand Tracking

This section deals with the problem of tracking of the regions previously detected by the adaptive skin color detector. Though using skin color is not a completely discriminative method which only detects hands, it is a necessary approach to be incorporated into our modules in order to reduce the amount of area in the image to process. More discriminative algorithms will be endeavored in the near future so the tracking is only performed on hands.

The task of tracking hands relies on the *Mean-Shift* algorithm, which is a robust method of finding local extrema in the density distribution of a data set. The Mean-Shift algorithm iterates to find the object center given its back-projection and initial position of search window. Back-projection algorithm consists in finding the pixels in the image which present features belonging to the distribution, described by a color histogram. The initial position of the search window is obtained by the skin detector algorithm in Section 3.2.2. The iterations are made until the search window center moves by less than the given value and/or until the function has done the maximum number of iterations. Usually, the mean-shift algorithm works for rectangular windows in an image. For tracking, the feature distribution to represent an object must be chosen first (object detection).

A related algorithm is the *Cam-Shift* tracker. It differs from the mean-shift in that the search window adjusts itself in size. In order to decide if a single tracker is following a wrong object, the distance between the current histogram (object model) and the initial one is computed. Whenever the difference between them is bigger than a certain percent, the tracker is considered to have lost the object, and this instance disappears from the list of tracked regions. On the other hand, any newly appearing region which is not yet tracked will generate a new instance of the tracker which will be independently followed.

So far, the input data which is provided to the tracker in the updating step corresponds to color information from the current frame in the video sequence. In a more advanced stage of the project, the previously described features will be incorporated into the tracking module, so the process is going to be more discriminative so only hands will be followed.

6 Description of the Software Developed in Task 3.1

This section will describe the set of general operations that have been implemented in the modules corresponding to the Task 3.1, in order to process the images and video sequences used throughout this project. The aim is to put together all the required elements of existing computer vision libraries as well as the set of implementation carried out by the author which will be necessary at any stage not only in the development of the present Task, but also in that of the rest of Tasks in WP3 which need to perform any analysis on visual information.

All the operations related to Task 3.1. have been included in a number of classes in C++. These classes make use of some types and functions already existing in the open-source computer vision libraries OpenCV and IVT (Integrating Vision Toolkit). OpenCV is a computer vision library originally developed by *Intel*. It is free for use under the open source *BSD* license. The library is cross-platform and it focuses mainly on *real-time* image processing. The IVT is an easy-to-use, platform-independent open source C++ computer vision library with an object-oriented architecture developed at the *Institute for Anthropomatics of the Karlsruhe Institute of Technology*. It offers a clean camera interface and a general camera model, as well as many fast implementations of image processing routines and mathematic data structures and functions. The IVT offers its own multi-platform GUI toolkit.

By constructing our set of classes, we obtain a tailored code that fits to our needs in a way that, on the one hand, it internally makes use of some of the operations already available in the previous libraries, while on the other hand, some of such operation has been modified when needed to be adapted to our requirements, but also some code has been created from the scratch. The advantages of this implementation are several. First, that of easing the combination of different modules and libraries, since it homogenizes the data and functions that different classes share each other. Secondly, it facilitates the use of already existing algorithms in other libraries, since these classes enclosure all the required variables and auxiliary data necessary for them to run, completing some of the implementation in OpenCV, which still do not make use of an object-oriented architecture in all its functions and libraries. Besides, it helps their integration with functionalities existing in the IVT library. Finally, this implementation increases the reliability of the code as it becomes more and more complex, and helps its integration as the number of modules increases.

Here, we describe the main functionalities of these classes:

- **CProcessVideoBase**: This class is the basis class to perform any process on a video sequence. Besides the initialization parameters, this class requires the use of element of the class **CProcessDataBase**, which houses all the variables required to process a frame: a pointer to a model and a pointer to the function which actually processes each single frame. A model consists in a pointer to an existing class, being it in OpenCV, ITV, or any other library, which is needed for keeping the information extracted from the image. Currently, existing models are background model, contour model, and feature extraction models. Instances of this class works with two video streams, that of the input, being it from a file or a camera, and that of the output into a file, besides providing functions for the visualization of the video streams. It is meant to be the basis class for developing any further implementation which integrates more processing modules.

- **CExtractFeatureBase**: This is the basis class concerned with the extraction of features on video frames. Specifically, the basis class is able to obtain the SURF features. Other derivative classes are **CExtractFeatureSIFT** and **CExtractFeatureHOG**, which implement the extraction of SIFT and HoG features. The main member functions of these classes allow the particular preprocessing of the image before obtaining their features, applying some basic algorithms to detect interest points, such as corners and star points detectors. The detection and extraction are divided into two steps, the computation of the interest points using the particular algorithm corresponding to each kind of feature, and the construction of the feature description. Besides these operations, there are some other functions which allow the clustering of features, using a k-Means clustering algorithm, and also their matching to another set of features, by means of a k-dimensional tree search.

- **CBagVisualWords**: This class implements the bag of visual words for the description of images. From a list of sets of features (the corpus), the algorithm extracts a vocabulary, a set of words (cluster representatives of all the features in the corpus) which will be posteriorly used to compute weights for each of the features representing a certain part of an image, that is, the bag of visual words. Other member functions in the class permit the use of different types of frequencies to count the presence or not of visual words, which are posteriorly used as vector weights, as well as the measures of distance between two BoVW.

- **CTrackerBase**: This class has been created as a basis class for detecting and tracking hands. The idea is creating an experimenting tool able to track hands and obtaining useful descriptions of the regions tracked which will be used posteriorly in other modules. It is not meant to be a final version of a hand tracker, but a working tool with which easily experimenting new approaches and, therefore, developing some improvements out of it. Currently, this class encompasses the skin color region detector and the Mean-Shift tracking algorithm, as well as the extraction of features in the regions which are tracked. A set of post-processing operations are also available, performing blob analysis on the detected segments to remove inadequate regions, and combining similar or close regions to form a single one.

Other auxiliary classes that were necessary to implement are:

- **CProcessParam**: This class is meant to carry any parameter necessary both for the initialization of any class as well as the utilization of any function. This way, all classes share the same parameter structure, and there is no local constant defined and lost within the code, which could later generate problems during the integration of different modules into one single piece of software.

- **OpenVideo**: This class manages the different streams from video files and from camera, opening, closing, reading, writing files, and visualizing frames. It also helps to use at the same time different formats belonging to the different libraries used here (OpenCV and IVT), and communicating their image buffers, which are expressed in different types of variables.

- **CProcessDataBase**: This class encompasses most of required variables needed in the processing of a frame, such function parameters, image buffer pointers, auxiliary images in different types and depths, pointer to a model, and also pointers to source and destiny image buffers. The idea is to put together in one single class all the necessary auxiliary structures assisting the processing of a video frame, and also being the basis for any further derivation of other similar classes which might be required in later developments of this software.

- **CClusterBase**: This is the basis class for implementing some clustering algorithms which can be used with the feature descriptors obtained using the member functions in the previous class **CExtractFeatureBase**, or called by some function in other classes of modules. Currently, k-Means and Expectation-Minimization (EM) algorithms are included in their member functions. The rest of member functions are concerned with the computation of cluster centers, and with labeling of samples according to the set of clusters obtained.

7 Work Progress and Conclusions

7.1 Progress towards Objectives

Low-level image and video processing objectives, namely, these of selection and construction of distinctive features, have been accomplished so far. By the use of OpenCV and IVT libraries, and the set of classes considered above, it is possible to carry out on video images operations as filtering, interest point extraction, skin color regions detection, and segment tracking. Despite, most of these operations rely on established techniques, this work has been indispensable in order to identify, adapt, and integrate those techniques that are best suited to the fulfillment of our objectives. In addition, putting together all the codes in a useful way so their use and posterior integration into other developments in WP3 will be easier, more robust, and less prone to errors, software bugs and failures.

With respect to the selection, construction, and extraction of distinctive spatio-temporal features, at the present moment it is possible to extract from videos sets of interest point and also features associated with them. It is also possible the construction of the aforementioned set of features on sets of predetermined points (grids, contours, regions, or a list of points), which has consisted in a deep modification of already existing feature extraction algorithms, and is thought to be effective in the posterior description of general hand movements and configurations.

In order to convert such frame descriptions into spatio-temporal features, a number of matching techniques have also been put at work so features from a certain video frame can be put in correspondence with those in the posterior frame. This approach will generate temporal correspondences of features by means of concatenating descriptors corresponding to analogous interest points in consecutive frames. Currently, these kind of descriptors are being tested to determine the most appropriate way to generate them.

The first attempt to determine a feasible and useful construction of the description of hand movement has been accomplished by means of bag of visual words, which is meant to be an effective way of describing any hand configurations. At present, this module, which is already implemented, is being tested for discriminative analysis of hands.

As part of the present Deliverable, a set of videos are provided that show the performance of some of the developments implemented so far within the Task 3.1. Specifically, these videos illustrate the functioning of the contour detection based on Canny algorithm (**contours.avi**), the adaptive skin color segmentation (**skin.avi**), the background/foreground segmentation carried on by the codebook algorithm (**background.avi**), and the extraction of the set of interest points (**features.avi**) corresponding to the SURF feature description of image patches.

7.2 Future Work

Following, we are stating the tasks which will be endeavored in the near future. Work must be carried out both in further testing the existing modules, the development of spatio-temporal features and in the implementation of discriminative tracking of hand movements.

There will be an implementation of new spatio-temporal features, such the Harris-type temporal generalizations in [5, 6]. This kind of features integrates at the same time temporal and spatial information so detection and description are performed at the same time in both dimensions. So far, there has not been access to the source code for such kind of feature description and either their implementation from the scratch or finding an alternative to them will be endeavoured in the forthcoming months.

The complete integration of all modules and code generated in Task 3.1. is to be done so far to a more extent that is done at present. This will consists in the combination of detection and tracking modules with those of feature extraction. Also the experimentation with different approaches in order to construct the distinctive description of hand movement with the already implemented set of features, as well as different interest point detection schemes must be carried out in order to determine the best approach.

Finally, the spatio-temporal feature extraction must be integrated into the baseline prototype in D.3.2. in order to provide these descriptors to other hand analysis modules which are able to perform a discriminative analysis of hand movements. On our regard, also more discriminative hand tracking techniques will be tried in order to improve the robustness and precision of hand tracking.

8 Conclusions

The current state of the tasks that have been developed corresponding to the Task 3.1. consists in the wide implementation and test of all the objectives related to the low-level image and video processing as well as the selection and construction of distinctive features for the analysis of hand movements. According to the current state of the developments corresponding to the Task 3.1, it must be stated that there are no significant deviation from the original work programme.

9 References

[1] *An adaptive real-time skin detector based on Hue thresholding: A comparison on two motion tracking methods*. Farhad Dadgostar, Abdolhossein Sarrafzadeh. Pattern Recognition Letters 27 (12), 1342-1352, 2006.

[2] *Discriminative human action recognition in the learned hierarchical manifold space*. Lei Han, Xianxiao Wu, Wei Liang, Guangming Hou, Yunde Jia. Image and Vision Computing 28, 836-849, 2010.

[3] *Vision-based hand pose estimation: A review.* Ali Erol, George Bebis, Mircea Nicolescu, Richard D. Boyle, Xander Twombly. Computer Vision and Image Understanding 108, 52-73, 2007.

[4] *Evaluating Bag-of-Visual-Words Representations in Scene Classification*, Jun Yang, yu-Gang Jiang, Alexander G. Hauptmann, Chong-Wah Ngo, International Multimedia Conference, Proceedings of the international workshop on Workshop on multimedia information retrieval, 197-206, Germany 2007.

[5] *Local Velocity-Adapted Motion Events for Spatio-Temporal Recognition*, I. Laptev, B. Caputo, C. Schuldt and T. Lindeberg; in *Computer Vision and Image Understanding*, 108:207-229, 2007.

[6] *Local descriptors for spatio-temporal recognition*, I. Laptev, T. Lindeberg. ECCV'04 Workshop on Spatial Coherence for Visual Motion Analysis, Springer Lecture Notes in Computer Science, Volume 3667. 91–103, 2004.

[7] *Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories*. S. Lazebnik, C. Schmid, J. Ponce. In Proc. Of 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, 2169-2178, 2006.

[8] *Disctintive Image Features from Scale-Invariant Keypoints*, D.G. Lowe. Int. Journal of Computational Vision, 60 (1), 63-86, 2004.

[9] *SURF: Speeded Up Robust Features*, H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Computer Vision and Image Understanding, vol. 110, No. 3, 346-359, 2008.

[10] *Histograms of Oriented Gradients for Human Detection*, Navneet Dalal, Bill Triggs, Int. Conf. On Computer Vision and Pattern Recognition, vol. 2, 886-893, 2005.

[11] Real-time foreground-background segmentation using codebook model, K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis, Real-Time Imaging 11 (2005): 167–256.

[12] *Learning OpenCV: Computer Vision with the OpenCV Library*. Gary Bradsky and Adrian Kaehler. O'Reily, September 2008.

10 Annexes

10.1 Background Segmentation

In this Task, the codebook background segmentation algorithm is employed as opposed to other more simple algorithms, such as background subtraction. In a general way, in order to perform background subtraction, a model of the background must first be learned. Once learned, this background model is compared against the current image and then the known background parts are subtracted away. The objects left after subtraction are presumably new foreground objects, faces and hands in our case. Normally, background is considered to be any static or periodically moving parts of a scene that remain static or periodic over the period of interest.

Unlike there are other quicker methods, which are mostly good for indoor static background scenes whose lighting does not change much, here we follow this by a codebook method that is slightly slower but can work in both outdoor and indoor scenes; it allows for periodic movements and for lighting to change slowly or periodically. This method is also tolerant to learning the background even when there are occasional foreground objects moving by, as it is likely in some environments such as offices or outdoor scenes.

Many background scenes contain complicated moving objects and also varying lighting. A method to deal with this would be to fit background statistical models to each pixel or group of pixels, but its disadvantage is the need for a great deal of memory. To get fairly close to the performance of adaptive filtering, the codebook algorithm takes inspiration from the techniques of video compression and attempt to just represent significant states in the background .

The simplest way to do this would be to compare a new value observed for a pixel with prior observed values, which are already modeled. If the value is close to a prior value, then it is modeled as a perturbation on that color. If it is not close, then it can seed a new group of colors to be associated with that pixel. The result could be envisioned as a bunch of blobs floating in a color space, each blob representing a separate volume considered likely to be background.

In practice, the choice of color space is done so that its axis is aligned with brightness, that is, the third component in *YUV*, *HSI*, or *HSV* color spaces. The reason for this is that, empirically, most of the variation in background tends to be along the brightness axis, not the color axis. The next step is to describe the background model. The blobs of the background model can be represented as Gaussian clusters. Nevertheless, there is a more efficient and simpler case, in terms of memory required and computational cost of determining whether a newly observed pixel is inside any of the learned boxes.

In this representation, the blobs are simply boxes with a learned extent in each of the three axes of our color space and the codebook is made up of such boxes that grow to cover the common values seen over time. If a value is too far away, then a new box is formed to cover it and likewise grows slowly toward new values. In the case of a background model, the codebook of boxes is learned that cover the three channels that make up our image at each pixel. This codebook method can deal with pixels that change levels dramatically and a foreground object that has values between the pixel values can also be detected.

In the codebook method of learning a background model, each box is defined by two thresholds (maximum and minimum) over each of the three color axes. These box boundary thresholds will expand (maximum getting larger, minimum getting smaller) if new background samples fall within a learning threshold above max or below min, respectively. If new background samples fall outside of the box and its learning thresholds, then a new box will be started.

In the background difference step, that is, when the image pixels are classified as belonging or not to the background model, there are acceptance thresholds. By using these threshold values, a pixel is said to be close enough to a maximum or a minimum box boundary and then counted as if it were inside the box. A second runtime threshold allows for adjusting the model to specific conditions. A situation this algorithm is not covering is the case of a moving camera, which is not considered due the static nature of the video data taken into account.

Summary of the codebook background segmentation algorithm:

- Learn a basic model of the background over a few seconds or minutes.
- Clean out stale entries (if the last frame to be learned).
- Adjust the thresholds to best segment the known foreground.
- Maintain a higher-level scene model.
- Use the learned model to segment the foreground from the background.
- Periodically update the learned background pixels.
- At a much slower frequency, periodically clean out stale codebook entries.

For a deeper description of this algorithm, please refer to [11, 12].

10.2 Adaptive Skin Detection

This algorithm is based on the extensive usage of hue cue of pixels for skin segmentation which provides one of the fastest methods for implementing a skin detector. Because it is a one dimensional color space, therefore, the skin color region can be specified by two global thresholds.

This part of the algorithm using a static skin detector, the global skin detector, can detect the actual skin pixels with reasonable rate requiring fer CPU instructions per pixels, that makes it a good candidate for real-time skin detection. However, falsely detected pixels from non-skin areas and from objects having similar hue values can appear in some situations in higher amounts than actual skin pixels, which makes it impractical for real-world applications.

To improve the results in real-wold applications, this algorithm also takes advantage of two facts. First, peaks in histograms (position, hight and width) are dependant on the image grabbing hardware and, therefore, the best results for a static algorithm can only be achieved by using the same kind of hardware in the training and detection steps. Secondly, manually changing the lower and upper bounds of the thresholds significantly improve the results, and the new thresholds are always inside of the boundaries of a global skin detector. Additionally, the skin information in the image is updated locally using motion features of the image, which means to improve the detection of the skin color through the time.

The skin detector algorithm has the following main steps:

- **Detecting in-motion skin pixels:** The next step is detecting the in-motion pixels of the image, using a motion detection algorithm, and filtering the detected pixels using the global skin detector. The output of this step are those pixels that with a higher probability belong to the skin regions of the image. Motion, in the present version of the algorithm is detected by frame difference in hue frames.

- **Updating the thresholds:** The pixels that were considered as moving pixels in the preceding step belonging to the user's skin are used for retraining the detector. In this implementation, a hue factor histogram is used as the base for calculating lower and upper thresholds for filtering the image in the next step. From the in-motion skin pixels, another histogram is extracted, and the second histogram is merged to the original histogram using an alpha mixing factor.

- **Filtering using adaptive skin detector:** For each frame, thresholds of the hue factor are recalculated such that they cover 95% (also an adjustable percent by the user) of the area of the new histogram. Afterwards, the pixels in the images are separated according their hue value with respect to these thresholds.

In summary, the basic rule to decide whether a pixel corresponds to skin or not encompasses the three cues before: hue, intensity, and motion, in the following way. For each pixel, if the intensity is inside a fixed interval and the hue factor lies belong the two thresholds, then, the image to be filtered is updated in case the hue factor lays inside the limits described by the adaptive thresholds, as explained in the *filtering* step. Among these pixels, those with a motion bigger than a certain threshold will be sent to update the histogram belonging to the moving pixels, as explained in the *detecting* step. Posteriorly, the hue histogram of the skin and the hue histogram corresponding to the motion image are updated and combined using a mixing equation in the *updating* step. The last step of the algorithm consists in a set of morphological operations aiming at the removal of disperse pixels, which occur because of the camera noise, color fluctuations and other noisy factors.

10.3 Feature Descriptors

Following, a brief explanation of these feature descriptors:

- **SIFT (Scale Invariant Feature Transform)**: This feature can robustly identify objects even among clutter and under partial occlusion, because this SIFT feature descriptor is invariant to scale, orientation, and affine distortion, and partially invariant to illumination changes. SIFT keypoints are defined as maxima and minima of the result of difference of Gaussian functions applied in scale-space (on the same image at different scales) to a series of smoothed and re-sampled images. Low contrast candidate points and edge response points along an edge are discarded. Dominant orientations are assigned to localized keypoints. These steps ensure that the keypoints are more stable for matching and recognition. The SIFT features are highly distinctive, relatively easy to extract, allow for correct object identification with low probability of mismatch and are easy to match against a (probably large) database of local features.

- **SURF (Speeded Up Robust Features):** It is a robust image detector and descriptor that is habitually used in tasks like object recognition and 3D reconstruction. It is partially related in concept to the SIFT descriptor, though the standard implementation is claimed to be several times faster than SIFT and, according to their authors, more robust against different image transformations than SIFT. SURF is based on sums of approximated 2D *Haar* wavelet responses and makes an efficient used of integral images. As basic image features it uses a *Haar* wavelet approximation of the determinant of Hessian blob detector.

- **HoG (Histogram of Oriented Gradients):** These feature descriptors are constructed by counting occurrences of gradient orientation in localized portions of an image. This method differs from other similar ones like SIFT in that it computes on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved performance. Essentially, the idea behind HoG descriptors is that local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions. The implementation of these descriptors can be achieved by dividing the image into small connected regions and for each one, computing the histogram of gradient directions for the pixels within the region. The combination of these histograms then represents the descriptor. The HoG descriptor maintains a few key advantages over other descriptor methods since it operates on localized regions, like invariance to geometric and photometric transformations, which appear in larger spatial regions.

10.4 Bag of Visual Words

Based on keypoints (local interest points) extracted as interest image patches, an image can be described as a bag of visual words and this representation has been frequently used in the classification of visual data.

Keypoints are salient image patches that contain rich local information about an image, which can be automatically detected and represented by descriptors. Keypoints are then grouped into a large number of cluster with other similar descriptors.

The sets vary in cardinality and lack meaningful ordering. This creates difficulties for learning methods (classifiers) that require feature vectors of fixed dimension as input. This is problem is addressed by using the vector quantization technique which clusters the keypoint descriptors in their feature space into a larger number of clusters using the *k-Means* clustering algorithm and encodes each keypoint by the index of the cluster to which it belongs.

By treating each cluster as a visual word that represents the specific local pattern shared by the keypoints in that cluster, we obtain a visual word vocabulary, which describes all kinds of such local image patterns. With its keypoints mapped into visual words, an image (or part of it) can be represented as a *Bag of Visual Words* (BoVW), or specifically, as a vector containing the (weighted) count of each visual word in that image, which is used as feature vector in the classification task.

10.5 Color-based Hand Tracking

The task of tracking hands relies on the *Mean-Shift* algorithm, which is a robust method of finding local extrema in the density distribution of a data set. This is a straightforward process for continuous distributions, which is essentially a hill-climbing algorithm applied to a density histogram of the data. For discrete data sets, this is a somewhat less trivial problem. The word robust is used here in its formal statistical sense, that is, Mean-Shift ignores outliers in the data. This means that it ignores data points that are far away from peaks in the data. It does so by processing only those points within a local window of the data and then moving that window.

The Mean-Shift algorithm iterates to find the object center given its back-projection and initial position of search window. Back-projection algorithm consists in finding the pixels in the image which present features belonging to the distribution, described by a color histogram. The initial position of the search window is obtained by the skin detector algorithm in Section 3.2.2. The iterations are made until the search window center moves by less than the given value and/or until the function has done the maximum number of iterations. The Mean-Shift algorithm is based on kernels to estimate probability density functions, where kernel is a weighting function used in non-parametric estimation techniques (e.g., a Gaussian distribution). When this change is zero, we are at a stable (though perhaps local) peak of the distribution, though there might be other peaks nearby or at other scales.

Usually, the mean-shift algorithm works for rectangular windows in an image. For tracking, the feature distribution to represent an object must be chosen first (object detection). Then, the Mean-Shift window starts to search over the feature distribution generated by the object, and finally computing the chosen feature distribution over the next video frame. Starting from the current window location, the mean-shift algorithm will find the new peak or mode of the feature distribution, which (presumably) is centered over the object that produced the color and texture in the first place. In this way, the Mean-Shift window tracks the movement of the object frame by frame.

The Mean-Shift algorithm runs as follows:

- Choose a search window.
- Compute the window’s center of mass.
- Center the window at the center of mass.
- Return to the second step until the window stops moving (it always will).

A related algorithm is the *Cam-Shift* tracker. It differs from the mean-shift in that the search window adjusts itself in size. If a well-segmented distributions is at hand (say face features that stay compact), then this algorithm will automatically adjust itself for the size of face as the person moves closer to and further from the camera. For tracking applications, the resulting resized box found on the previous frame should be used as the window in the next frame.

Despite these two algorithms are able to track distributions of any kind of features, the object model in our case consists in a color histogram which is initialize out of each region detected by the color skin detector, which first localizes the window the Mean-Shift algorithm will follow. This histogram is computed for each search window at every frame in the sequence and used to find the object by back-projection to the current frame.

In order to decide if a single tracker is following a wrong object, the distance between the current histogram (object model) and the initial one is computed. Whenever the difference between them is bigger than a certain percent, the tracker is considered to have lost the object, and this instance disappears from the list of tracked regions. On the other hand, any newly appearing region which is not yet tracked will generate a new instance of the tracker which will be independently followed.